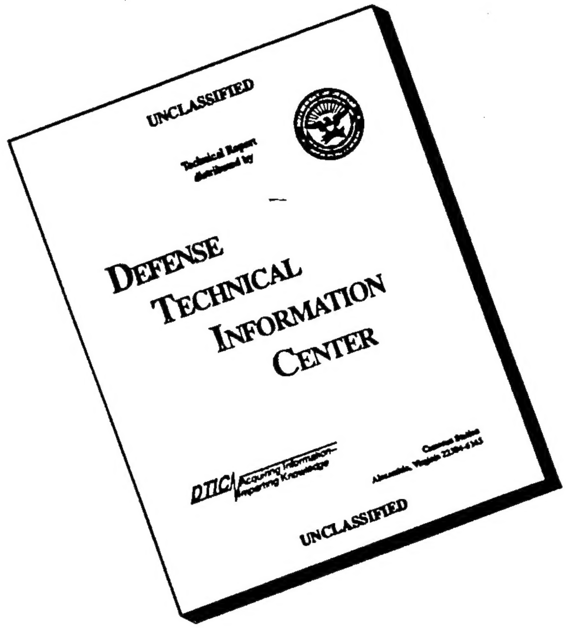
Second Consortium Reengineering Workshop: Approaches to Reengineering for Information Systems

SPC-96044-CMC
Version 01.00.04

June 1996

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Second Consortium Reengineering Workshop: Approaches to Reengineering for Information Systems

SPC-96044-CMC

Version 01.00.04

June 1996

Produced by the
SOFTWARE PRODUCTIVITY CONSORTIUM

SPC Building
2214 Rock Hill Road
Herndon, Virginia 22070

CONTENTS

This page intentionally left blank.

PREFACE

These proceedings are based on the presentations that were made at the *Second Consortium Reengineering Workshop: Approaches to Reengineering for Information Systems*, held at the Software Productivity Consortium in Herndon, Virginia on December 4 and 5, 1995.

This page intentionally left blank.

1. INTRODUCTION

1.1 OVERVIEW

This document contains the results of the second Software Productivity Consortium (Consortium) Reengineering Workshop, held at the Consortium on December 4 and 5, 1995. This introduction provides general information on the workshop and a description of how the remainder of the document is organized.

1.2 WORKSHOP OBJECTIVES

The objectives of the workshop were to gather reengineering researchers and practitioners from industry and academia to discuss directions in reengineering. The workshop sought to compare approaches to reengineering information systems, including the current state-of-the-practice of reengineering of legacy systems, data and process reengineering, product lines, and object technology in reengineering.

1.3 THE WORKSHOP PROCESS

Thirty two people attended the workshop. Their names, organizations, and addresses are given in Appendix A. Attendees were invited to submit position papers prior to the workshop, make a presentation at the workshop, or both. Eight attendees submitted position statements, but one cannot be included because of copyright restrictions. Thirteen attendees made presentations.

1.4 ORGANIZATION OF THIS DOCUMENT

This document is organized as follows:

- Section 2 contains position papers submitted to the workshop.
- Section 3 contains copies of the slides presented at the workshop.
- Section 4 contains the results of the workshop.
- Appendix A contains a list of attendees.
- Appendix B contains the final workshop agenda.

1.5 TYPOGRAPHIC CONVENTIONS

This document uses the following typographic conventions:

Serif font .... General presentation of information.

Italicized serif font .... Publication titles.

Boldfaced serif font .... Section headings and emphasis.

2. POSITION STATEMENTS

This section contains the position statements submitted by workshop attendees. The statements are arranged alphabetically by authors' names. The following table lists the authors and the titles of their position statements (empty entries in the Title column means the authors did not supply a title for their position statement).

Shawn Bonner and Clement McGowan	Bridging the Gap Between Business Process and Software System Reengineering
Dan Juttelstad	
Julia McCreary	Reengineering Large Legacy Systems Case Study: Internal Revenue Service
Boris Mutafelija	
Anne Rose	Reengineering User Interfaces: A Case Study
Karen White	
Mark Wilson	

This page intentionally left blank.

Name: Shawn A. Bohner and Clement L. McGowan

Company: The MITRE Corporation

Division:

Address: 7525 Colshire Drive, MS-Z667

McLean, Virginia, 22102-3481

Telephone: Bohner (703) 883-7354 McGowan (703) 883-7099

FAX #: (703) 883-6991

E-Mail: bohner@mitre.org mcgowan@mitre.org

Title: Lead Scientist Principle Engineer

(INCLUDED) I will be sending you a position statement by Nov. 30.

(INCLUDED) I am interested in making a reengineering presentation at the workshop. My topic is on: Bridging BPR and Software Systems.

Title of Presentation: Bridging The Gap Between Business Process and Software Systems Reengineering

Abstract: The bridge between business process and information systems reengineering is all too often missing from the roadmap of reengineering efforts. When process and system engineers get to this transition, they discover a rickety old bridge with steep terrain on either side of a wide chasm. Recognizing this dilemma, we developed the Business Reengineering for Information Technology (BRIT) approach that systematically transitions from business process to information systems engineering. BRIT is designed to handle a wide range of reengineering factors including: "best practices," COTS applications, non-standard business processes, and change situations ranging from continuous improvement to radical restructuring. This proven approach is described with relevant examples of its applications.

Describe your experience with reengineering projects to date.

The authors have held leadership positions in a wide range of software system reengineering efforts both in industry and the public sector. Most of the efforts over the past 5 years have been in the public sector and have focused primarily on modernizing legacy systems. The following are examples of these experiences.

The first example is a system that was originally developed in the 1960's and enhanced over twenty years. It was written in COBOL with data managed without the support of a commercial DBMS. The system was housed on an obsolete platform and software documentation was minimal. Using a combined top-down/bottom-up approach logical data models, business rules, and the like were captured and documented. These models were redeveloped for a new

open systems architecture and specified for an relational database management system implementation.

The next example is a system that was originally developed as a prototype and deployed to the field in an operational environment. Without the requisite software development discipline and software documentation, software changes were difficult at best and many times impossible. Technology improvements led to platform obsolescence and ultimately the requirement that the system be modernized. Since the software was not well developed, a porting effort was not feasible. Instead, a targeted capture effort was used to identify, qualify, extract, refine/redocument, and wrap algorithms and functions determined to be useful in the new system. The rest of the project followed a more traditional development approach concentrating on a flexible architecture and reuse of the captured components.

Another example is a system that was originally developed in the late 1988's in COBOL with data managed using a hierarchical database (tuned for speed). The system was developed on an aging platform and the database architecture was deemed to be inflexible for the types of changes that the system was subject to. Using a combined top-down, bottom up approach the logical data models, business rules, and the like were captured. The database was restructured to take advantage of relational database features and redeveloped for a new open systems architecture.

What are the problems with reengineering approaches you have tried or observed?

Many of today's system modernization efforts are sparked by business process reengineering (BPR) efforts. A new process levies new requirements that the existing software system cannot support without significant changes. While new processes resulting from the BPR effort describe business level requirements, there exists a sizable gap between these requirements and the requirements needed to modernize the supporting software systems.

Approaches that have a high reliance on tool technology to reverse engineer software artifacts from source code are subject to considerable risk. Most tools advertised to accomplish software reverse engineering do not capture the requisite information needed to understand the system. Some tools are able to capture physical representations of the program and data design, but there is considerable work in transforming these representations into logical models since much of the semantic information is not conveyed in the source code.

What are you looking for in reengineering solutions, methods, and tools?

A trend that started a few years ago in the information systems engineering community was the merging of business engineering and information systems engineering concepts. We see evidence of this in CASE tools where process modeling tools are being integrated with information systems modeling tools. We are looking for integrated approaches that enable issues from both domains to be addressed.

While today's tools and methods offer some help in modeling processes and

systems, technology to support reverse engineering is still lagging behind. Since information conveyed to the computer in the form of a program does not contain direct logical design or architecture information, evidence of this information must be inferred from patterns in the code and available documentation. Therefore, we should be looking for knowledge-based approaches for discovering the information necessary to reverse engineer existing systems. Just as importantly, we need to be implementing ways that new systems can be represented so that they can be readily reverse engineered in the future.

This page intentionally left blank.

Telephone: __401-841-4581__________________________________

FAX #: __401-841-2031_____________________________________

E-Mail: __juttelstad@code22.npt.nuwc.navy,mil______________

Title: ___Elect Eng____

(X) I will be sending you a position statement by Nov. 30.

( ) I am interested in making a reengineering presentation at the workshop. My topic is on: ______________________.

Position Paper

Dan Juttelstad
NUWCDIVNPT
Code 2253, 1171-2
Newport RI

NUWCDIVNPT Code 2253 has been addressing re-engineering of software for reuse for 4 years.
In that time a process has been developed that integrates commercial off the shelf (COTS) tools
for performing Domain Analysis, Software Assessment, Software Re-Engineering, and Software
Resuse Repository support.

The primary objective is to develop the Undersea Software Domain Reuse Repository. This
repository is intended to contain resusable software components that meet the requirments of the
undersea domain model. The software components come from existing systems software and
new development software. The components are evaluated for quality with respect to reuse and
re-engineered, if necessary, for incorporation into the reuse library.

The primary issue in performing re-engineering for reuse is the definition of metrics associated
with the re-engineering process and determing cost versus value added. It is difficult to predict
the value added by re-engineering and determineing if it is cost effective.

This page intentionally left blank.

Reengineering Large Legacy Systems
Case Study: Internal Revenue Service

Background

The Internal Revenue Service (IRS) story of maintaining (and attempting to replace) aging, piecemeal legacy systems is one which is familiar to many institutions. In the process of evaluating reengineering as an enabling technology, we discovered critical issues broader than the mechanical definitions of methods and tool technology. Rather, the organizational framework became the essential element; for example, identifying business objectives, performing a portfolio analysis, planning technology transition.

The IRS has been engaged in reengineering projects and studies since 1990. Three reengineering efforts were completed in Fiscal Year 1992, identifying specific IRS opportunities to utilize this technology in support of implementing Tax Systems Modernization, a massive effort planned to replace the tax systems software over a 10-year period. One project made a high-level evaluation of IRS systems to identify candidates for reengineering, based on an assessment of IRS needs and business objectives. A second project identified tools that could support the objectives identified in the first project. The third project was a prototype to demonstrate technical issues and solutions. Two smaller projects were completed in FY93, with an enterprise-wide assessment of current systems scheduled to begin in FY94. Funding for that project was postponed until FY95 and substantially reduced. Projects currently underway include a Year 2000 Project and four reuse/reengineering projects in support of new development. Throughout this period, an aggressive effort to integrate reengineering principles into the software development environment and market their benefits in light of organizational goals has met cultural as well as business challenges.

Definition

Software reengineering is defined as an enabling technology, supporting redevelopment in various strategic and tactical ways. It refers to a variety of techniques and tools employed in support of the process of using components from existing systems to improve the current system, whether that improvement includes a complete redesign and rewrite of code, a transition to a new equipment/ software platform or a simple redocumentation of current systems.

Concerns

Objective-driven

Reengineering is objective-, or goal-, oriented. There are so many applications to which the technology can be applied (platform migration, data translation, language upgrade, redocumentation) that the definition of the term is dependent upon the use being made of it in a particular application. This aspect of the "discipline" needs to be considered when creating a framework or outlining a life-cycle for redevelopment.

Cultural Barriers

The marketing of reengineering as a supporting and enabling technology for the process of software development is essential to its acceptance in the organizational culture unfamiliar with it. Organizations unfamiliar with reengineering techniques and possible benefits may resist, based on an assumption that this effort will detract from and divert resources from organizational development goals or that reengineering is counter to the "new development" efforts. These barriers must be addressed early and often. They will, no doubt, continue to be a part of the environment into which reengineering will be integrated. Management support, once garnered, can be essential in keeping the momentum going against cultural resistance.

Transition Issues

Transition and the orderly retirement and replacement of legacy systems with newly developed or redeveloped systems is one of the most critical issues facing the IRS today. Most replacement scenarios do not have a clean one-to-one mapping of functions and data to the systems they are replacing. The management of identifying functions and preparing the systems and data being replaced is a part of the reengineering discipline that is essential to a successful Tax Systems Modernization effort.

Conclusions

It has been said by many, but the essential ingredients to successful implementation of an enabling technology like reengineering are organizational. Organizational needs must be identified and assessed. Clearly, not every organization requires every technical solution available through reengineering. A supportive management group must be identified. A SMALL pilot project which will result in a production solution to a real problem, and has a high likelihood of success, needs to be identified and recognized. The results of that project need to be publicized, using that success to integrate reengineering in the software development process elsewhere in the organization. And when the pilot is not a success, or the organization changes direction, rendering the transition plans out-of-step, the organizational needs must be revisited to bring the reengineering solutions to bear on those with a good return-on-investment. Recent industry "hype" has resulted in high expectations being held by management, followed by disappointment and a sense that the technology has "nothing useful to offer" . A plan should be developed for implementing those aspects of reengineering that are sensible for the organization, based on recognized business goals and reasonable supporting technology.

Return-Path: <brewer@medusa.Software.ORG>
Received: by Software.ORG (/\==/\ Smail3.1.25.1 #25.19)
id <m0tKTOP-0001PAC@Software.ORG>; Tue, 28 Nov 95 12:00 EST
Received: by medusa.Software.ORG (/\==/\ Smail3.1.25.1 #25.10)
id <m0tKTOS-000039C@medusa.Software.ORG>; Tue, 28 Nov 95 12:00 EST
Date: Tue, 28 Nov 1995 12:00:31 -0500 (EST)
From: Gerry Brewer <brewer@Software.ORG>
Subject: 2nd Reengineering Workshop (fwd)
To: facemire@Software.ORG
Message-ID: <Pine.3.89.9511281225.H23269-0100000@medusa>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII

---------- Forwarded message ----------
Date: Tue, 28 Nov 95 11:55:00 EST
From: Mutafelija, Boris <MUTAFBO@gateway.grumman.com>
To: "'smtp:brewer@software.org'" <brewer@software.org>
Subject: 2nd Reengineering Workshop

Gerry,

As per your announcement please register me for the 2nd SPC Reengineering Workshop.

Name: Boris Mutafelija
Company: Northrop Grumman Corp.
Division: Data Systems and Services
Address: 2411 Dulles Corner Park
Herndon, VA 22071-3430
Phone: (703) 713-4174
Fax: (703) 713-4103
e-mail: borism@gateway.grumman.com
Title: Information Systems Technologist

Attached below is my "Position Statement".

Northrop Grumman Data Systems and Services Division External Information Systems Business Unit is extending its standard organizational software process to include reuse and reengineering. We often encounter requirements to reengineer a large body of legacy software and transition this reengineered software into a solution required by the customer. Typically, our customers require that reengineered systems satisfy additional requirements, such as enhancements to existing software, integration of reengineered legacy systems with new applications (frequently COTS software), etc.

Problems that we experience can be classified as:

1 - System engineering problems

? development of complete solutions that will include reengineered legacy software, newly developed software, and COTS software (forward- and re-engineering combined)
? addition of new capabilities, including increased reliability, maintainability, transportability
? process/methodology addressing multistep reengineering (i.e. reengineering that covers code conversion, data conversion, database conversion, rehosting, adding new capabilities, etc.)
? integration of reengineered legacy code with COTS software
? testing strategies

? transitioning (to a new system)

2 - Reengineering problems

? reengineer vs. reverse engineer vs. restructure: when is each one appropriate? how do they play together?
? reengineer and enhance (there are additional requirements to be satisfied)
? convert from one language (most frequently COBOL) to another (most frequently Ada)
? convert from hierarchical to relational database
? convert from mainframe to client-server; in addition, convert code from one language to another, and from one database format to another (multistep reengineering)

3 - Implementation problems

? system analysis/synthesis tools
? need for tool repositories
? system testing tools
? reengineering tools (rehosting, translating, conversion)

What we are looking for in reengineering solutions, methods and tools:

? reengineering process definition (hopefully related to ESP or GSEP)
? methodology for analyzing and synthesizing systems that contain reengineered legacy systems, have additional requirements, and may include COTS software (in order to satisfy all new requirements)
? reengineering taxonomy (classification) and implementation of such taxonomy to practical problems (similar to E. Chikofsky s paper in IEEE Software)
? development of test strategies (including test case generation) for such systems
? tools for analyzing, synthesizing and then testing such systems (forward- and re-engineering tools integrated into one tool-set (through a repository?))

? Project management aspects:
  ? estimation
  ? planning and scheduling
  ? controlling
  ? quality assurance and configuration management

? Process engineering aspects
  ? process descriptions
  ? life-cycle selection (waterfall, incremental, evolutionary)
  ? reengineering risk analysis (similar to R. Arnold s paper)

? Effectiveness
  ? cost/benefit models for each reengineering approach (e.g. conversion, reverse engineering, rehosting, mixture)
  ? when NOT to reengineer?

Re-engineering User Interfaces: A Case Study

Anne Rose

Human-Computer Interaction Laboratory
Center for Automation Research
University of Maryland
College Park, MD 20742
rose@cs.umd.edu
http://www.cs.umd.edu/projects/hcil

November 29, 1995

The Human-Computer Interaction Laboratory (HCIL) is currently under contract with the Maryland Department of Juvenile Justice (DJJ) to make recommendations for redesigning the user interface of their information system, ISYS. ISYS, Information System for Youth Services, is a terminal based system used to support the case processing of approximately 50,000 referrals of delinquent youth behavior. It is built around a centralized IDMS database that is running on an IBM mainframe located at the Annapolis Data Center. ISYS is used by about 600 DJJ employees in various offices and facilities across the state.

During our first year, we evaluated ISYS, proposed short term recommendations, and developed prototypes for NISYS, the next generation ISYS. We employed several techniques to learn, assess, and evaluate, ISYS including reading the documentation, performing 22 field visits, attending training sessions, getting our own hands-on experience, and administering the Questionnaire for User Interaction Satisfaction (QUIS) [2][6]. QUIS was developed by the HCIL to quantitatively evaluate the strengths and weaknesses of user interfaces. In consultation with DJJ, the QUIS was tailored to address specific issues of concern to DJJ and administered to 332 employees. The mean rating for ISYS, out of 9, was 5.1.

The field visits provided us with valuable insight about ISYS, and about the functioning of DJJ in general. A typical visit consisted of an overview from a supervisor, observing users performing routine tasks, and discussing what they liked and disliked. We refined our observation techniques and proposed an applied ethnographic method for redesigning user interfaces [5].

Based on our findings, we proposed 28 short term recommendations to improve the ISYS interface while NISYS is being developed. Our recommendations focused on:

- making system access easier,
- improving data accuracy,
- making information retrieval easier,
- increasing the usefulness of the system, and
- improving user satisfaction.

We provided a rough estimate of the payoff vs. the implementation cost for each recommendation. DJJ's initial response was to take action on all 28 recommendations. However, because of internal restructuring only a few recommendations have been implemented to date.

We also proposed three NISYS prototypes in response to the needs discovered during the evaluation:

- LifeLines, which uses time lines to display a youth's history with DJJ on one screen [4],
- the DJJ Navigator, which helps manage individual workloads by displaying different user views, and
- the Information Visualization & Exploration Environment (IVEE)[1], a generic tool that can be used to visualize, explore, and make queries on DJJ datasets [1].

We demonstrated these prototypes to 60 DJJ personnel and made revisions based on their comments. Overall, their feedback was very positive. One challenge we have found is how to get constructive feedback. For many DJJ employees, ISYS is their only computer experience so they find it difficult to provide constructive criticism. They seem to like our prototypes too quickly simply because it looks better than what they have now. Providing rough sketches, that don't look like finished applications, or providing alternative designs, might be the solution.

We have also been working with Cognetics, Corp., our subcontractor, who is working in conjunction with DJJ, to prepare the request for proposal (RFP) for NISYS. The Cognetics Design Methodology (CDM) is being used for this process [3]. We are currently working on the functional requirements. The NISYS project is serving as an exercise to test the practicality of CDM. CDM is being modified as new needs are discovered.

REFERENCES

[1] Ahlberg, C., Wistrand, E., IVEE: An Information and Visualization & Exploration Environment, *Proceedings of IEEE Visualization '95 (Atlanta, Georgia, October 1995)*, 66-73.

[2] Chin, J., Diehl, V., and Norman, K., (1988), "Development of an instrument measuring user satisfaction of the human–computer interface", *Proceedings of CHI'88—Human Factors in Computing Systems*, ACM, New York, 213–218.

[3] Kreitzburg, C., (1995), "Managing for Usability: The Cognetics Design Methodology", Cognetics, Corp., Princeton Junction, NJ.

[4] Plaisant, C., Milash, B., Rose, A., Widoff, S., Shneiderman, B., (1995), "LifeLines: Visualizing Personal Histories", to appear in *Proceedings of CHI '96*, ACM, NY.

[5] Rose, A., Shneiderman, B., Plaisant, C., (1995), "An Applied Ethnographic Method for Redesigning User Interfaces", *Proceedings of DIS '95 (Ann Arbor, Michigan, August 1995)*, ACM, NY, 115-122.

[6] Vanniamparampil, A., Shneiderman, B., Plaisant, C., and Rose, A., (1995), "User Interface Reengineering: A Diagnostic Approach", CAR-TR-3459, University of Maryland, College Park, MD.

[1] IVEE was developed by Christopher Ahlberg and Erik Wistrand of Chalmers University, Sweden. It is based on earlier research by HCIL. URL: http://www.cs.chalmers.se/SSKKII/ivee.html

Registration for SPC Reengineering Wkshp (fwd) *11/27/95 8:44:59 AM*

```
Return-Path: <brewer@medusa.Software.ORG>
Received: by Software.ORG (/\==/\ Smail3.1.25.1 #25.19)
 id <m0tHaeD-0001PHC@Software.ORG>; Mon, 20 Nov 95 13:08 EST
Received: by medusa.Software.ORG (/\==/\ Smail3.1.25.1 #25.10)
 id <m0tHaeD-0000AuC@medusa.Software.ORG>; Mon, 20 Nov 95 13:08 EST
Date: Mon, 20 Nov 1995 13:08:52 -0500 (EST)
From: Gerry Brewer <brewer@Software.ORG>
Subject: Registration for SPC Reengineering Wkshp (fwd)
To: facemire@Software.ORG
Message-ID: <Pine.3.89.9511201339.H18964-0100000@medusa>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
```

---------- Forwarded message ----------
Date: Mon, 20 Nov 1995 13:00:09 -0500
From: Karen White <krwhite@smtpgate.read.tasc.com>
To: brewer@software.org
Subject: Registration for SPC Reengineering Wkshp

Name: Karen R.J. White

Company: TASC

Address: 55 Walkers Brook Drive
Reading, MA 01867

Telephone: 617-942-2000 x. 2654

FAX #: 617-942-7100

E-Mail: krwhite@tasc.com

Title: Project Leader

( ) I will be sending you a position statement by Nov. 30.

POSITION STATEMENT IS INCLUDED BELOW.

( ) I am interested in making a reengineering presentation at the workshop. My topic is on:

POSITION STATEMENT:

I have been involved with the enhancement and maintenance of so-called legacy software systems for the past 17 years. During that time I have seen systems translated from one language to another, re-structured to provide better performance, and completely re-built. Within the immediate past history, I have participated in efforts that ranged from the reengineering/re-hosting of a user interface associated with a legacy system, to the development of a strategic plan for

Registration for SPC Reengineering Wkshp (fwd)   11/27/95 8:44:59 AM

reengineering of an application, to the reengineering of a "mission critical" (but not embedded!) system.

The problems we encountered were primarily associated with the "reverse engineering" activities and managing the user's expectations about the final product. The project management (both customer & user) held the assumption that one can successfully reengineer an application by analysing only the existing software, a piece at a time. We failed in conveying to them the benefit associated with having the current support staff participate in the reengineering project. (It should be noted that support of the legacy system was provided by another contractor & that they were the users of the new system; the customer was a DoD dept.) A comprehensive reverse engineering project, including development of models of the undocumented legacy system, should have been completed before we started the forward engineering; there would have been fewer surprises at the end. Or, the project should have been treated as a "scrap the old & let's build a new from scratch" with a full-blown requirements analysis phase.

My interests are:
(1) Approaches to reengineering user interfaces INDEPENDENT of the rest of the application;
(2) Connecting business process modelling, reverse engineering and forward engineering. What methods exist (don't exist) that allows a business manager to see in a straightforward fashion how the current system does or does not support the business process and how the proposed system will.
(3) Management of a reengineering project; how does one identify the risk areas, does one break the project up into reverse engineering projects and then forward engineering projects; how does COTS integration "mess up" the picture; where are the logical milestones for "go, no-go" decisions

UNIVERSITY OF MARYLAND AT COLLEGE PARK

OFFICE OF TECHNOLOGY LIAISON • GRADUATE STUDIES AND RESEARCH

QUESTIONNAIRE FOR USER INTERACTION SATISFACTION ("QUIS™")

The most powerful computer is a computer which people will use; similarly, software programs must meet the approval of the end user to be effective. Measuring and understanding user reactions to computer software is important to many who are creating new services and programs, evaluating older versions, or making choices between similar products for certain applications. While the evaluation of a system's accuracy is fairly straightforward, the assessment of the user's satisfaction with the human-computer interface is a subjective and complex question.

A multi-disciplinary team of researchers at the University of Maryland at College Park (UMCP) has developed an instrument which evaluates user satisfaction with the human-computer interface aspect of other software packages and computer systems. The Questionnaire for User Interaction Satisfaction ("QUIS™") includes a paper version as well as a computerized questionnaire which assesses users' attitudes and subjective satisfaction with a system, especially the users' evaluation of the human-computer interface. "Although a system may be evaluated favorably on every performance measure, the new system may not be used very much if the user is dissatisfied with the system and its interface," said Dr. Kent L. Norman of the UMCP Department of Psychology and the Human-Computer Interaction Laboratory (HCIL).

The QUIS™ covers four major areas: Screen, Terminology and System Information, Learning, and System Capabilities. Within each area, several issues are rated on a nine-point scale, with guides such as Barely Legible...Very Legible; Confusing...Clear; Difficult...Easy; Complex...Simple. The wide range of topics includes the computer's noise level, helpfulness of reference materials, even screen sequencing. The user is able to rate any computer program using QUIS™, thus producing a reliable evaluation of the interactive workstation. According to Dr. Benjamin Shneiderman of the Computer Science Department and Director of the HCIL, "The QUIS™ does two things. It taps the overall subjective reaction of a user to an on-line computer system, and it is a diagnostic of the strengths and weaknesses of a system. It assesses such things as satisfaction with the display of graphics, readability, reliability, understandability, and other features." Please see reverse for additional specifications.

4312 KNOX ROAD • COLLEGE PARK, MARYLAND 20742 • (301) 405-4209 • FAX: (301) 314-9871

The QUIS™: Questionnaire for User Interaction Satisfaction

Developed in the Human/Computer Interaction Laboratory at the University of Maryland by Kent Norman and Ben Shneiderman, the QUIS™ is one of the only instruments for assessing user evaluations of interactive workstations that has proven reliability and validity. It has been standardized over a number of applications and research studies. It is now being used in the field by a number of usability and research labs in both government and industry.

The QUIS™ assesses 6 factors of overall reactions to the system and 21 components that contribute to usability. The QUIS™ can be used in its current form or modified to meet particular research needs.

The QUIS™ is available in a paper version and two on-line versions (a Windows™ version and a Macintosh™ Spinnaker Plus™ version).

QUIS™ Site License Information

A site license for commercial use of the QUIS™ is now available for $750. A reduced fee ($200) is available for academic/non-profit use. The licensing package includes the following:

- Two copies of the QUIS™ paper version with the right to make an unlimited number of copies for use at one site.
- Copies of all HCIL publications pertaining to the use of the QUIS™.
- The Windows™ version of the QUIS™ on a 3.5 inch floppy disk with the right to use copies at your site.
- The Macintosh™ Spinnaker Plus™ stackware version of the QUIS™ on a 3.5 inch disk with the right to use copies at your site.
- Run-time versions of Macintosh Spinnaker Plus.

The site license gives authorization for unlimited use of the QUIS™ at one site. You may copy the entirety of the questionnaire, parts of it, or revise it for use in evaluation of commercial software/hardware for usability testing, research, and development.

For technical information on the QUIS™, contact:

Dr. Kent L. Norman
Department of Psychology
University of Maryland
College Park, MD 20904
(301) 405-5924

To receive the licensing package, contact:

Ms. Carolyn A. Garrett
Office of Technology Liaison
4312 Knox Road
University of Maryland
College Park, MD 20742
(301) 405-4210

Systems Reengineering Position Paper for Second SPC Reengineering Workshop

Mark L. Wilson

Naval Surface Warfare Center DD/WO
mlwilso@relay.nswc.navy.mil
301-394-5099

We, in the Navy tactical community, are concerned primarily with real-time, safety critical, mission critical, complex systems. Many of these systems are written in CMS-2 programming language with at least some and often substantial amounts of assembly code. They run on Navy military standard computers such as UYK-7, UYK-43, UYK-20, UYK-44, and AYK-14.

Most of these systems were designed with memory or other architectural constraints which no longer apply. Thus thorough reengineering requires consideration of the design rationale which may not be explicit in the existing documentation. Moreover, there may be timing or other relationships which only become apparent during the most thorough system test.

Driving factors for reengineering may include a desire to: avoid hardware obsolescence, increase performance to accommodate new or enhanced requirements, reduce software maintenance costs, reduce hardware procurement costs, reduce maintenance costs, and take advantage of current software design practices and tools. In essence the goal is rehosting or retargeting to improve performance, reduce cost, and reduce development time. A secondary goal is to try, where practical, to further reduce cost and development time through reuse.

One of our short term goals is the ability to efficiently transform highly hardware dependent legacy systems into moderately hardware independent open systems. Our long term goal is the ability to effectively transform complex legacy systems into systems that can evolve gracefully over time. Evolution will typically involve increased requirements, new processor hardware, new display technologies, and integration with additional systems. It may also involve reuse within or across systems and domains, totally new requirements unanticipated during design, partition into parallel or distributed architectures, combination into fewer more powerful processors, new human computer interfaces, networks to replace point-to-point communication, or translation from one language into another.

Some of the techniques we see as useful are a layered approach to system and software design, automatic or semiautomatic language translation, and graphical aids to software and system understanding. It may also be useful to have econometric models to guide the decisions of when and how to reengineer.

We at NSWC have addressed a number of these issues. Among them we have helped develop and test CMS-2 to Ada translation tools, an Assembler to CMS-2 translator, graphical aids to software understanding, and have worked on populating several domains with legacy components. Most of this work was funded by ONR, some by SBIRs, and some by the JLC/CRM/RRFWG. We also organized and sponsored a series of Systems Reengineering Technology Workshops - the last two of which were held in Monterey, California; and a Reengineering Focus Group at the First and Second Workshop on Engineering Systems in the Twenty-First Century (WES21).

This page intentionally left blank.

3. SLIDES PRESENTED

This section contains copies of the slides used in presentations at the workshop. The slides are arranged in chronological order (see Appendix B).

This page intentionally left blank.

SOFTWARE PRODUCTIVITY CONSORTIUM

Product Line Engineering

Jeff Facemire
703-742-7189

facemire@software.org
http://www.software.org

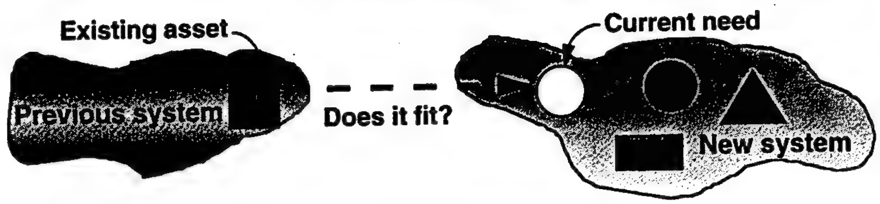
SOFTWARE PRODUCTIVITY CONSORTIUM

Product Lines

- Product Lines - A collection of (existing and potential) products that address a business area
- Recently mandated by Lloyd Mosemann (SAF/AQK) for the Air Force to do Domain Engineering to product lines
- Mosemann sited SPC's product line approach as implemented in the Navy/STARS program
- Other examples sited: PRISM and CARDS

SOFTWARE PRODUCTIVITY CONSORTIUM

Difficulties in Using Legacy Assets

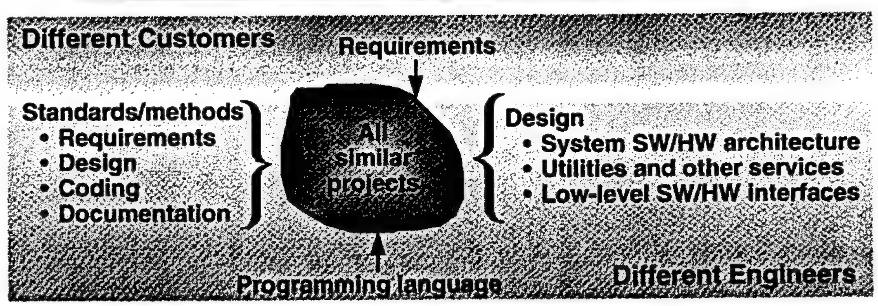


- A poor fit (actual or apparent) increases risk/cost of:
  - Recognizing the opportunity
  - Finding asset to reuse
  - Adapting asset to fit in new system
  - Verifying fit in new system

Variation among projects threatens a good fit.

SOFTWARE PRODUCTIVITY CONSORTIUM

Potential Sources of Variation



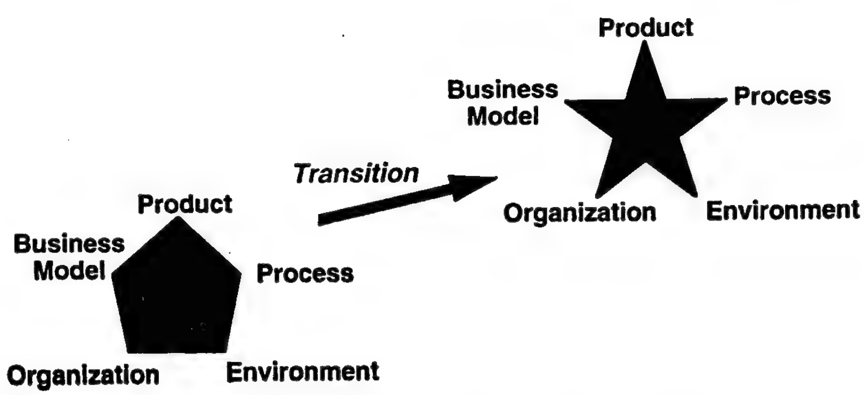
- **Legitimate** variations arise from different customer needs.
- **Incidental** variations arise in project management or technical approach and viewpoint.
- **Legitimate** and **incidental** variations interact to create complex variations among similar projects.

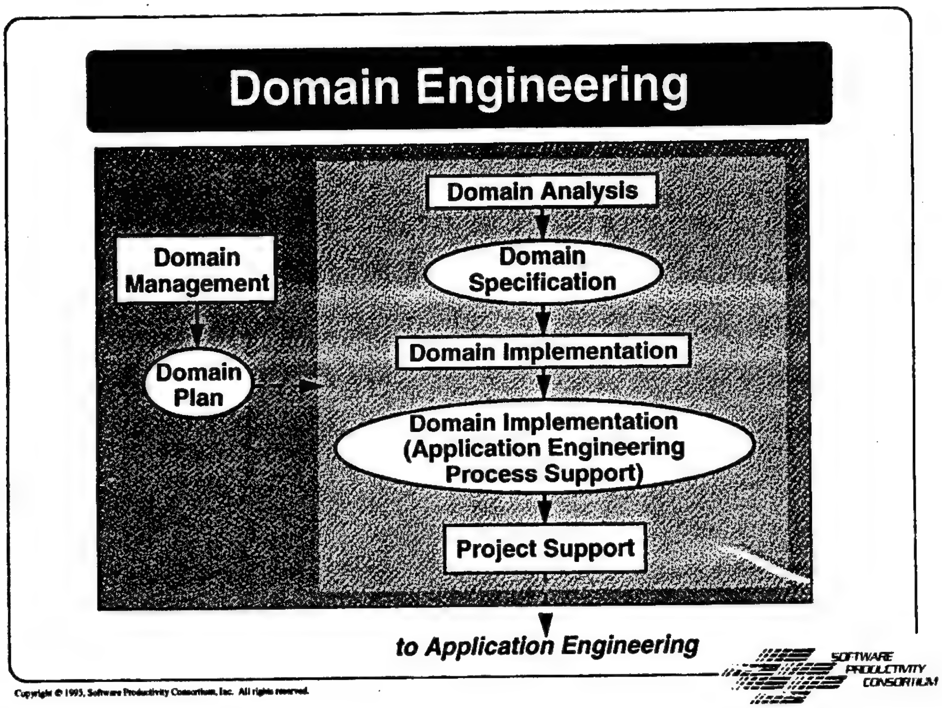
SOFTWARE PRODUCTIVITY CONSORTIUM

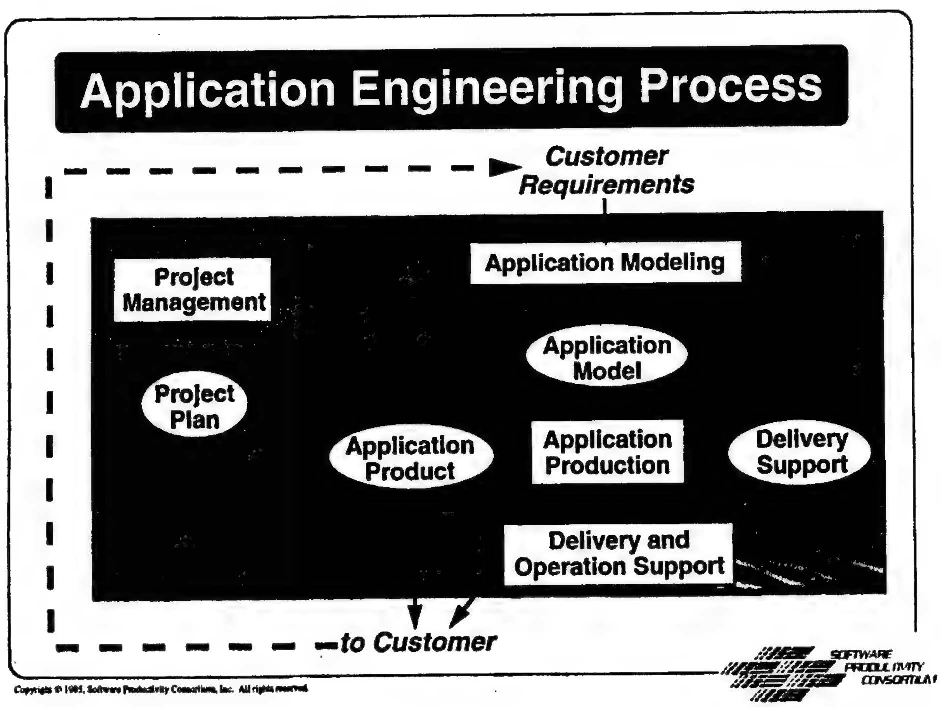
Relationship to Reengineering

- SPC's product line approach has been performing many aspects of conventional reengineering (organizational, process and product improvement)
- Reverse engineering can help to:
  - Determine commonalities and abstraction in legacy code
  - Capitalize on existing assets for use in the product line
- Product line engineering with or for reengineering will likely use both top-down and bottom-up analysis

Adoption Process

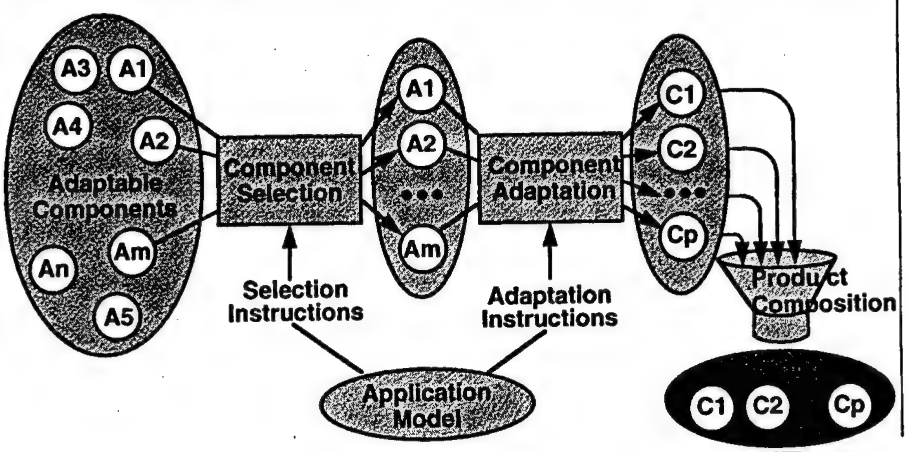



Domain Engineering
Domain Analysis
Domain Management
Domain Specification
Domain Implementation
Domain Plan
Domain Implementation (Application Engineering Process Support)
Project Support
to Application Engineering
SOFTWARE PRODUCTIVITY CONSORTIUM


Application Engineering Process
Customer Requirements
Application Modeling
Project Management
Application Model
Project Plan
Application Product
Application Production
Delivery Support
Delivery and Operation Support
to Customer
SOFTWARE PRODUCTIVITY CONSORTIUM

Result of Domain Engineering

•Application production is mechanical and, possibly, automated



Product Line Engineering

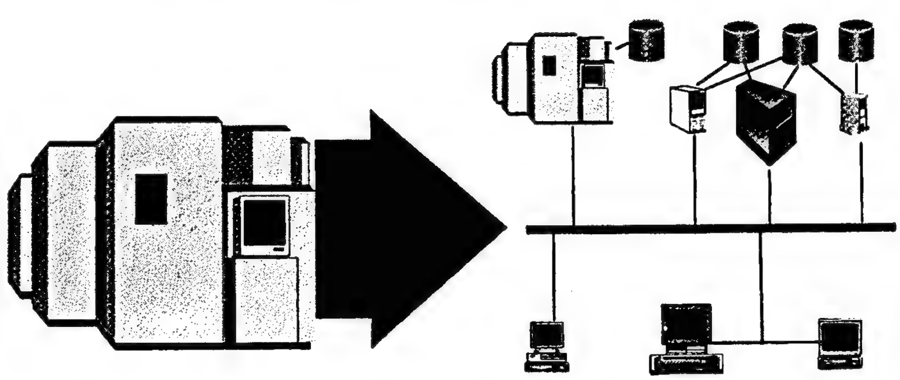
- Engineering product/component families and an associated production process to optimize support for a defined business area.
- Concern with reuse focused on given organization's business area
- Addressing variabilities via adaptability of product/components (including requirements, architecture, tests, etc.)
- Borrows/integrates other reuse technologies
- Examples: Consortium's Synthesis, GCSS/CASS, NSWC/RNTDS, Toshiba

Business and Technology Trends

- Business Trends:
  - Changing government role
  - Cost reduction
  - Cycle-time reduction (time-to-market improvement)
  - Incremental product changes
  - Information explosion
  - Market globalization
  - Rapid market changes
- Technology Trends:
  - Architecture-based composition of systems (application generation)
  - Business process reengineering and engineering process improvement
  - Distributed development
  - Distributed systems (client/server)
  - Integrated product and process development (multifunction teams)
  - Multimedia
  - Object-oriented development
  - Software-intensive systems

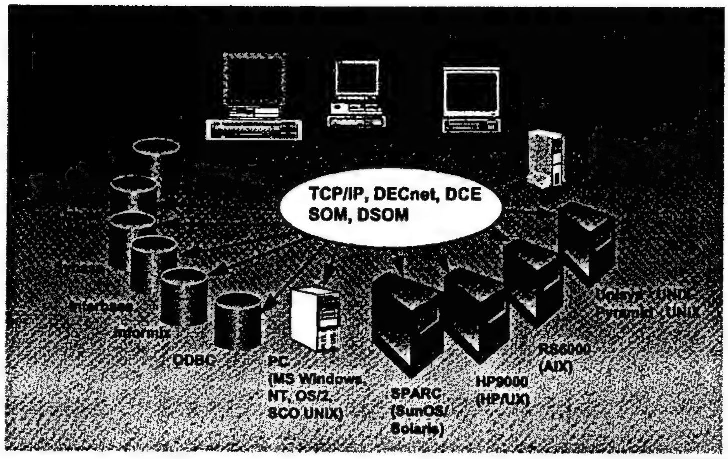
SOFTWARE PRODUCTIVITY CONSORTIUM

Mainframe to Geographically Distributed Client/Server



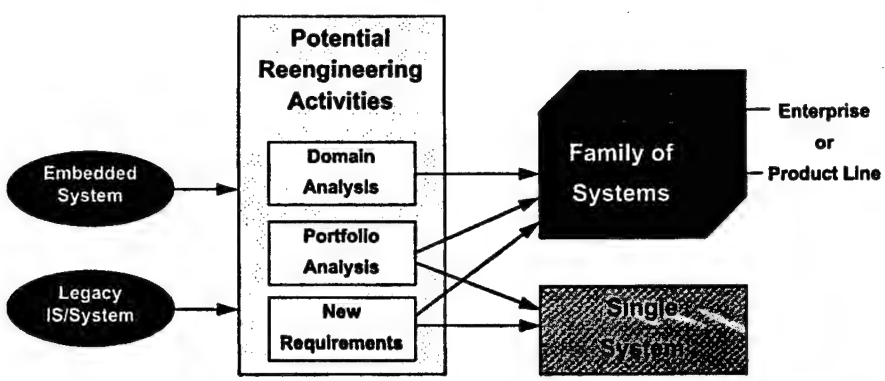
SOFTWARE PRODUCTIVITY CONSORTIUM

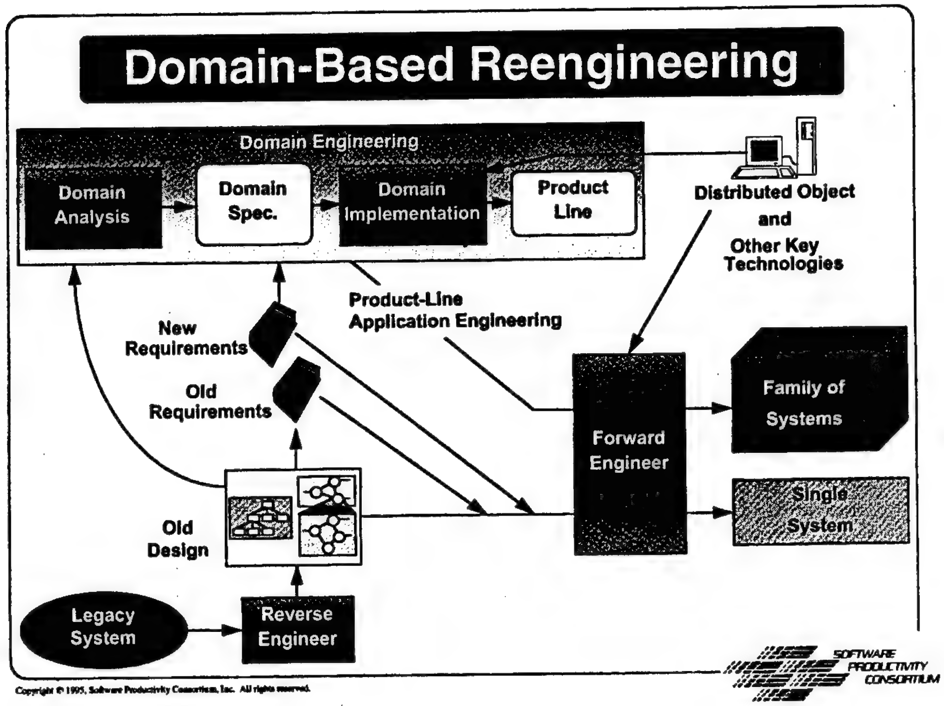
A Much More Versatile But Complex Physical Infrastructure

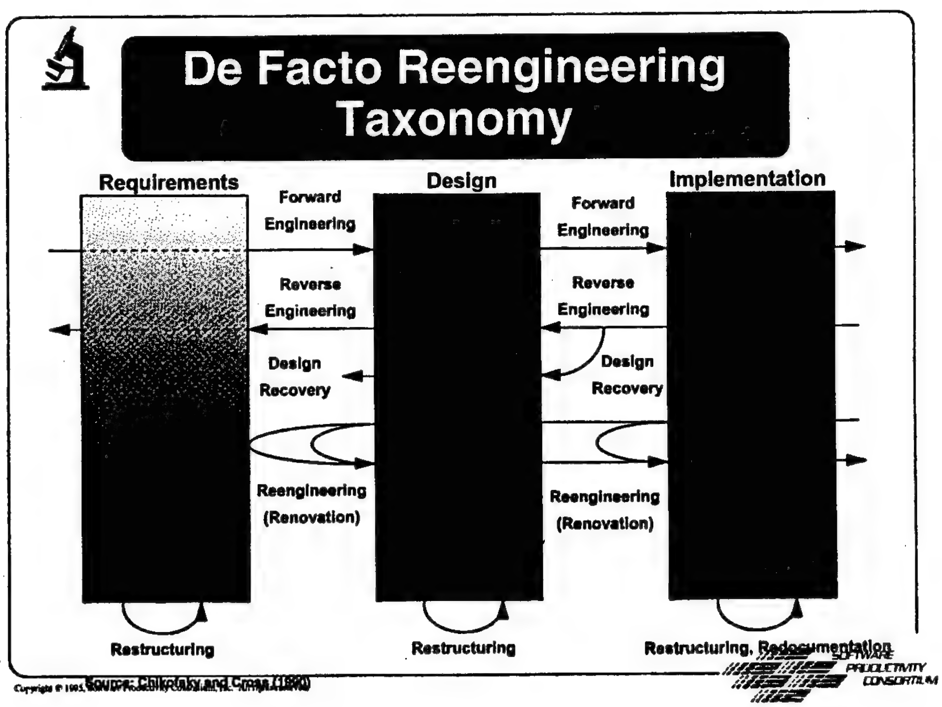


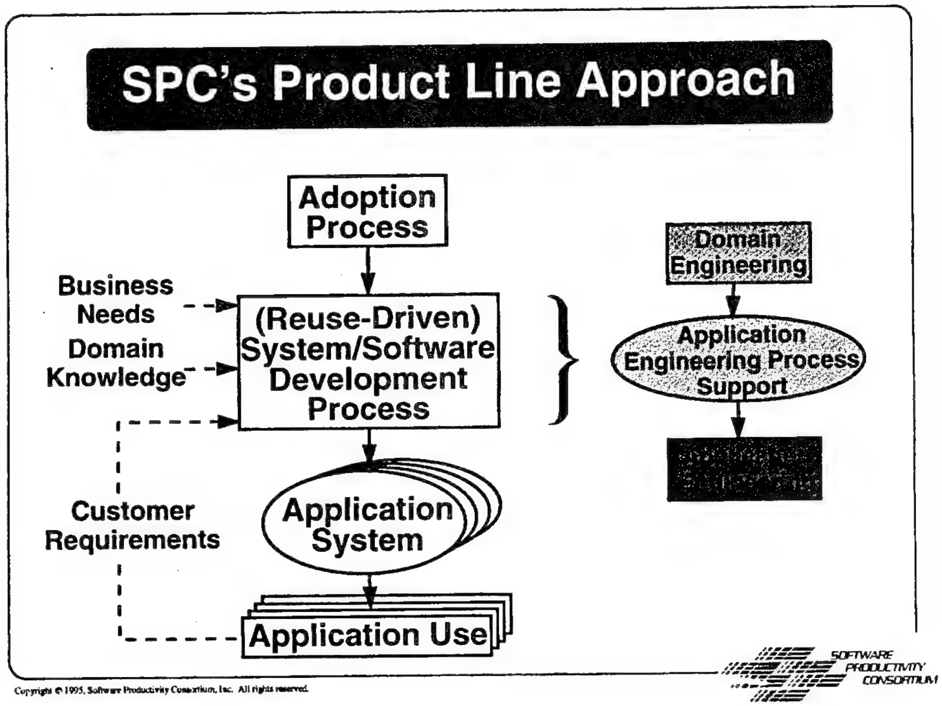
Reengineering of Information Systems

Family of Systems versus Single System




Domain-Based Reengineering
Domain Engineering
Domain Analysis
Domain Spec.
Domain Implementation
Product Line
Distributed Object and Other Key Technologies
Product-Line Application Engineering
New Requirements
Old Requirements
Old Design
Forward Engineer
Family of Systems
Single System
Legacy System
Reverse Engineer
SOFTWARE PRODUCTIVITY CONSORTIUM


De Facto Reengineering Taxonomy
Requirements
Design
Implementation
Forward Engineering
Forward Engineering
Reverse Engineering
Reverse Engineering
Design Recovery
Design Recovery
Reengineering (Renovation)
Reengineering (Renovation)
Restructuring
Restructuring
Restructuring, Redocumentation
Source: Chikofsky and Cross (1990)
SOFTWARE PRODUCTIVITY CONSORTIUM


SPC's Product Line Approach
Adoption Process
Business Needs
Domain Knowledge
(Reuse-Driven) System/Software Development Process
Customer Requirements
Application System
Application Use
Domain Engineering
Application Engineering Process Support
SOFTWARE PRODUCTIVITY CONSORTIUM
Copyright © 1995, Software Productivity Consortium, Inc. All rights reserved.

This page intentionally left blank.

MYTHS AND REALITIES

Defining Re-engineering for a Large Organization

Sandra Yin (ISM:TM:S)

Julia McCreary (ISD:I:SE)

Internal Revenue Service

1111 Constitution Ave.

Washington, D. C. 20224

Myths of R^3

Reverse and Re-engineering are synonymous

Re-engineering soils pure top-down effort

Old programs - nothing to salvage

Re-engineering is fully automated

Single CASE tool - solution for new development

Buy a CASE tool -

- Infrastructure not essential
- Work process - no need to examine
- Organizational readiness will just happen
- Process improvement is peripheral

Wishful thinking makes it so

5

Realities of R^3

Establish Clear Objectives

No Silver Bullet

Embrace Business Re-engineering Early

Get Experienced Guidance

Phased Change

Balance R&D with short term Return-on-Investment

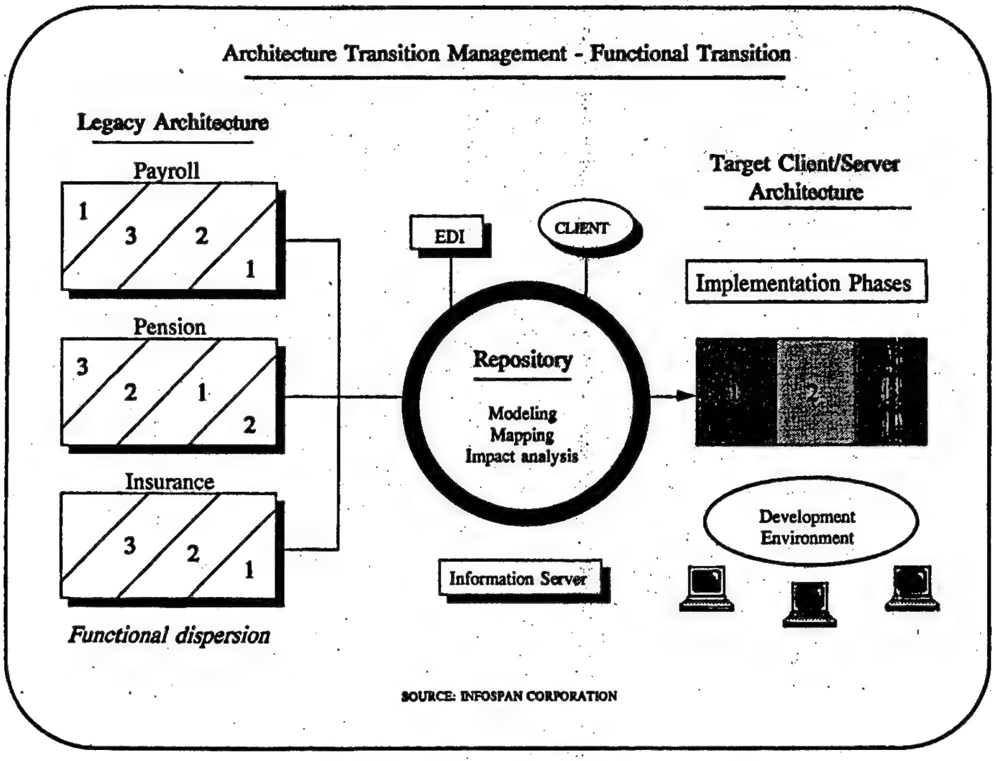
Gather Case Study Results

Transition is a major issue

38

Transition

- Cross References from Old to New Data and Processes
  - Conversion Routines
  - Synchronize Change Control

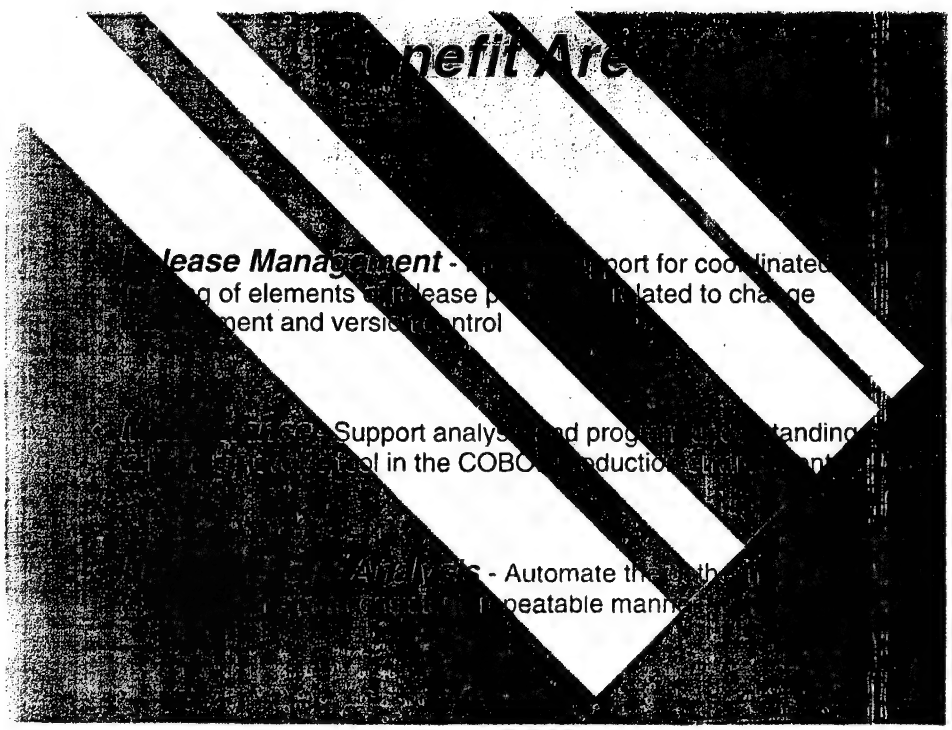


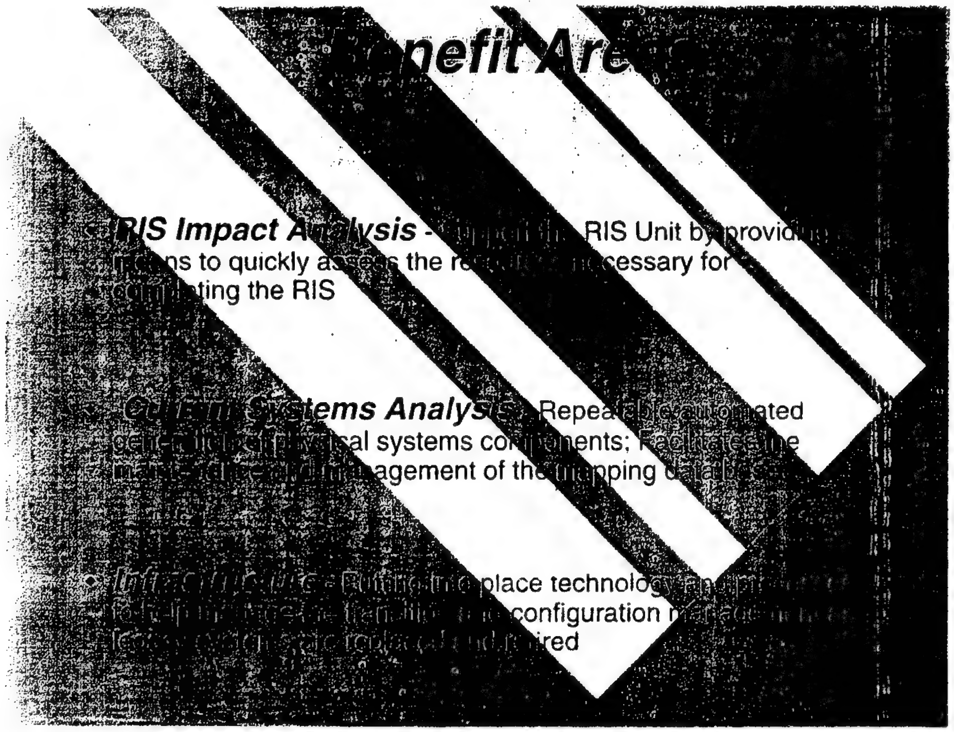
Technical Opportunities

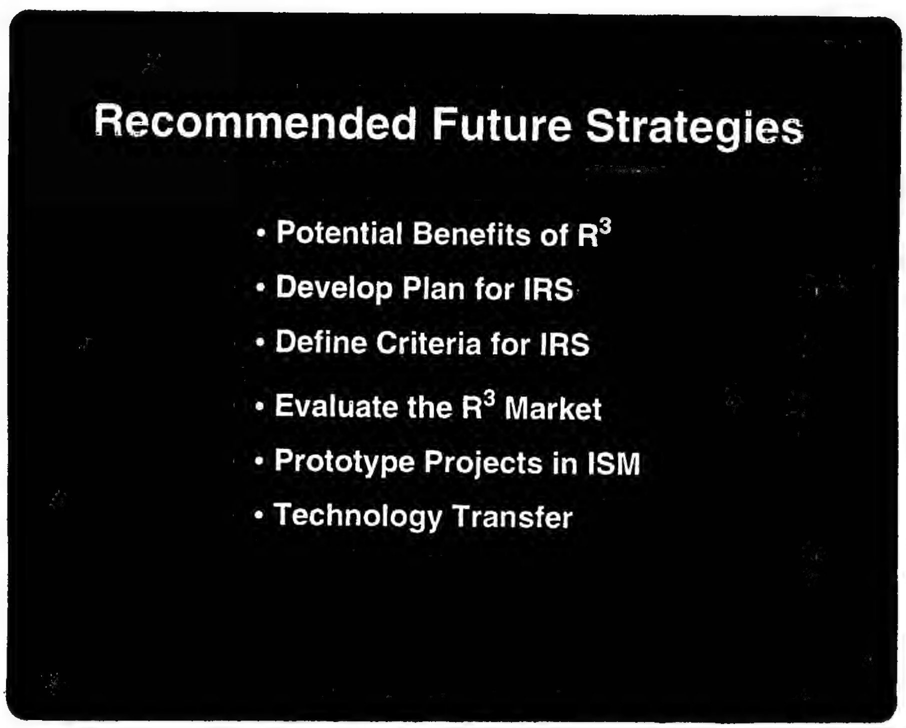
- Reuse
- System rationalization (data & process)
- Functional enhancement
- Technology platform conversion
- Technology redesign
- Re-documenting
- Restructuring

12


Looking for R^3 Realities
Establish objectives
Identify opportunities
Identify tools and method
Support transition strategy
Target implementation
6


lease Management -
ort for coo
g of elements
ease
ated to ch
ent and versi
ntrol
Support analy
d pro
in the COBO
ductio
- Automate t
peatable man


RIS Unit
ns to quickly
essary for
ing the RIS
ems Analy
al systems co
agement of th
place technolo
configuration
ired


Recommended Future Strategies
• Potential Benefits of R3
• Develop Plan for IRS
• Define Criteria for IRS
• Evaluate the R3 Market
• Prototype Projects in ISM
• Technology Transfer

This page intentionally left blank.

FUSION OF DOMAIN ENGINEERING AND REUSE WITH LEGACY CODE

Noah Prywes
University of Pennsylvania
and
Computer Command and Control Company
Philadelphia, PA 19103, PA

Outline

- PROCESS:
  Find legacy software components and fit them into a domain architecture

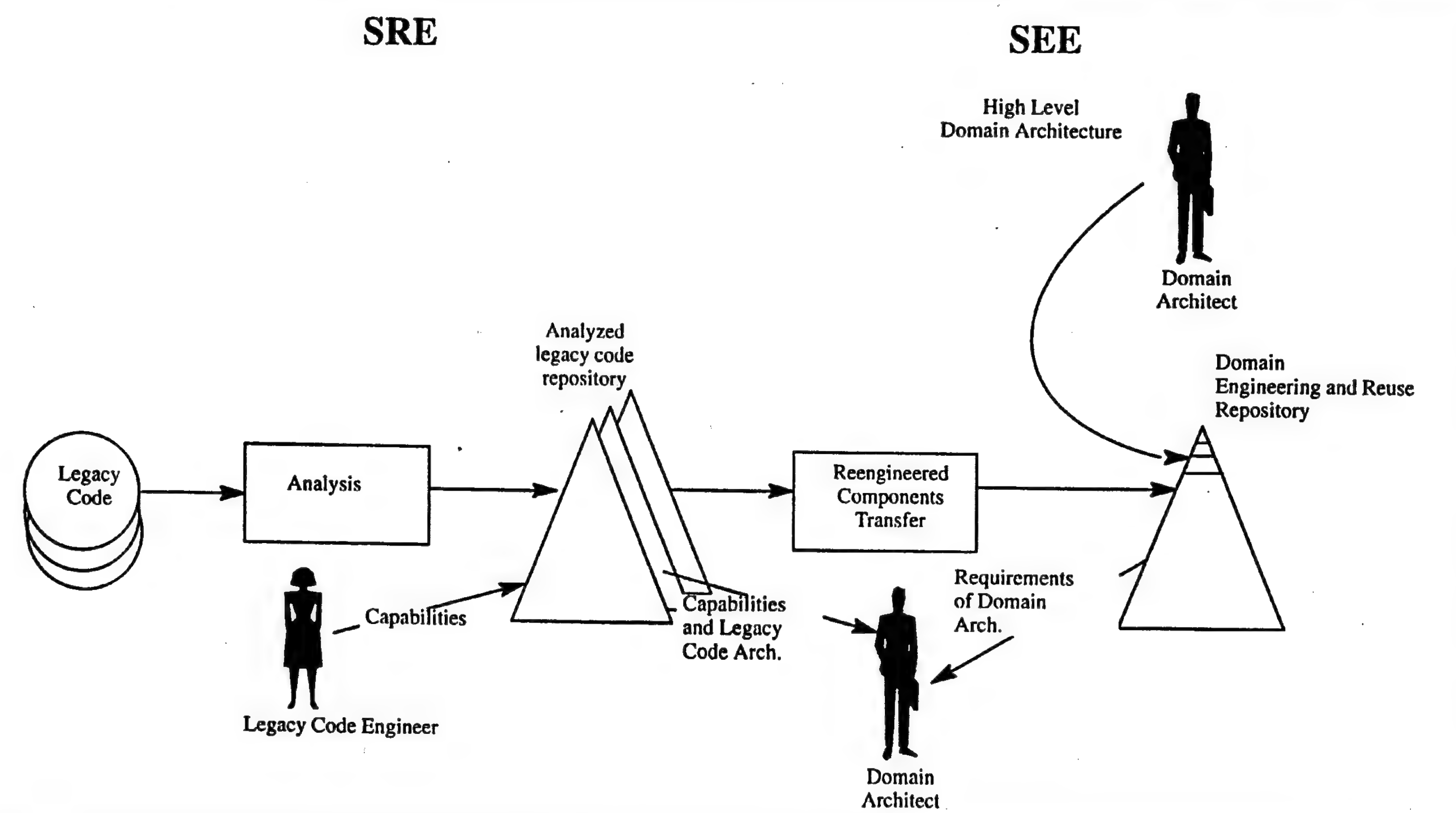
- ENABLING TOOLS FOR:
  Find and fit legacy components
  — CCCC's Software Reengineering Environment (SRE)
  Domain Architecture
  — Boeing's Software Engineering Environment (SEE)

- EXPERIENCE: NAWC–TSD
  Domain: Air Vehicle Training Systems (AVTS)
  Legacy: Propulsion Components of T–34, T–44 Trainers

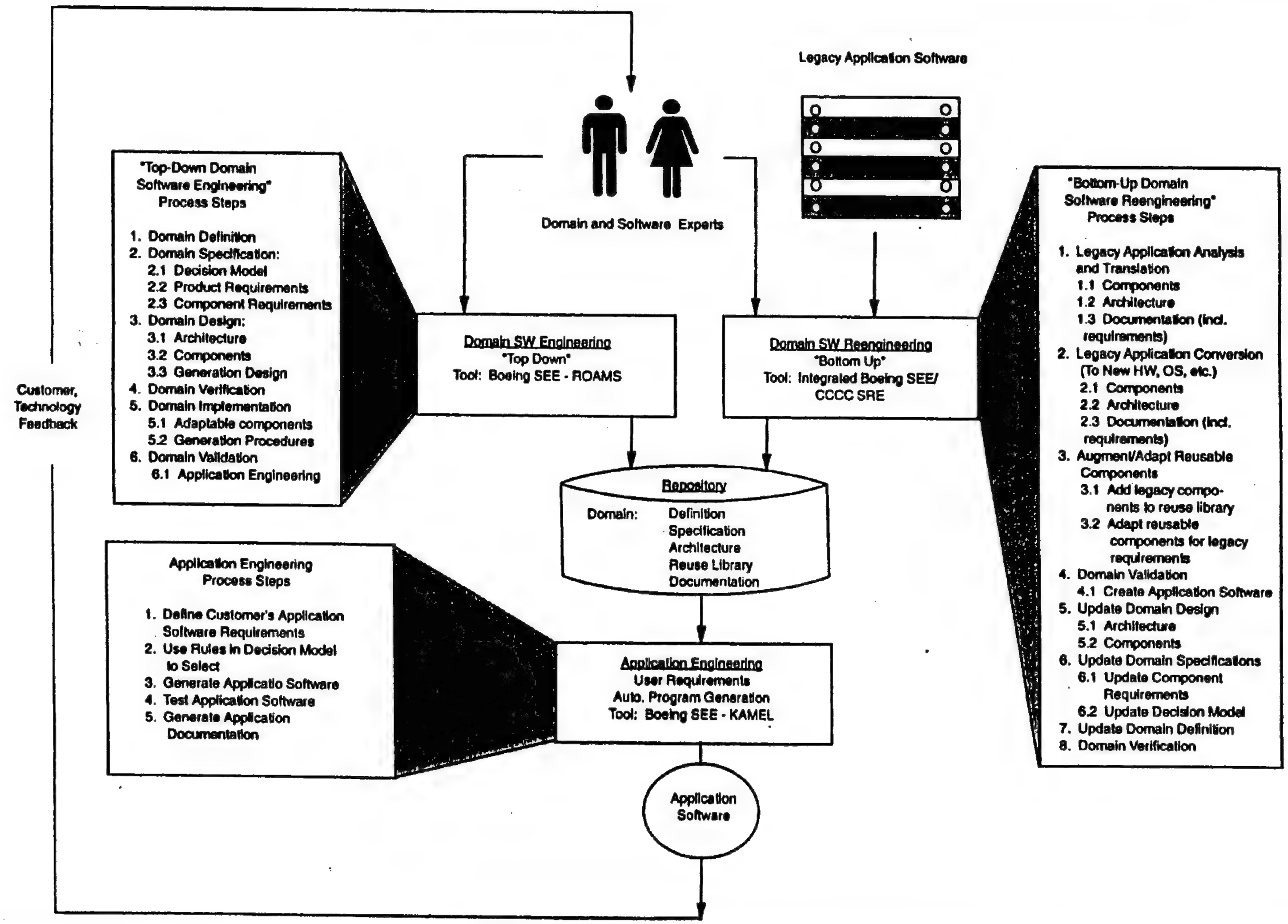
- FUTURE TOOLS:
  Select legacy components with least interfaces
  Test selected components
  Fit selected component into domain architecture

PROCESS:
FIND LEGACY COMPONENTS AND FIT THEM INTO DOMAIN ARCHITECTURE

Process Information Flow



Augmented Methodology Process and Steps



ENABLING TOOLS

Nodes and Edges in SRE Analyzed Legacy Code Repository

Nodes: software entities

1. At the lowest level: statements
2. Hierarchical Software Units (SWU) contain statements, data, and COTS with documentation.

Edges (Relationships between nodes):

1. Scope: between parent SWU and its children SWUs, in order of precedence; between block statements and their constituents
2. Memory: between a variable reference and its declaration
3. Type: between type of variable and its declaration
4. Call: between procedure/function declaration and its caller
5. Message: between message call and its destination task entry point
6. Context: between with/use and respective package, etc.

Other types of edges may be added.

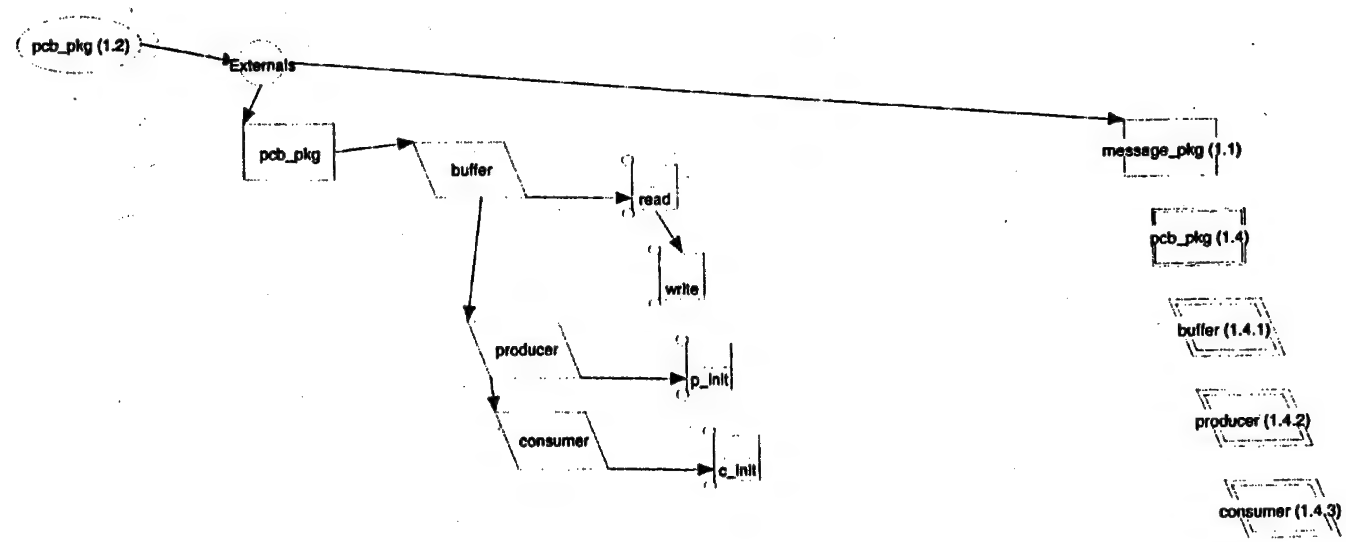
Defining SRE Software Units

Rules for discovery of architectural Software Units (SWU):

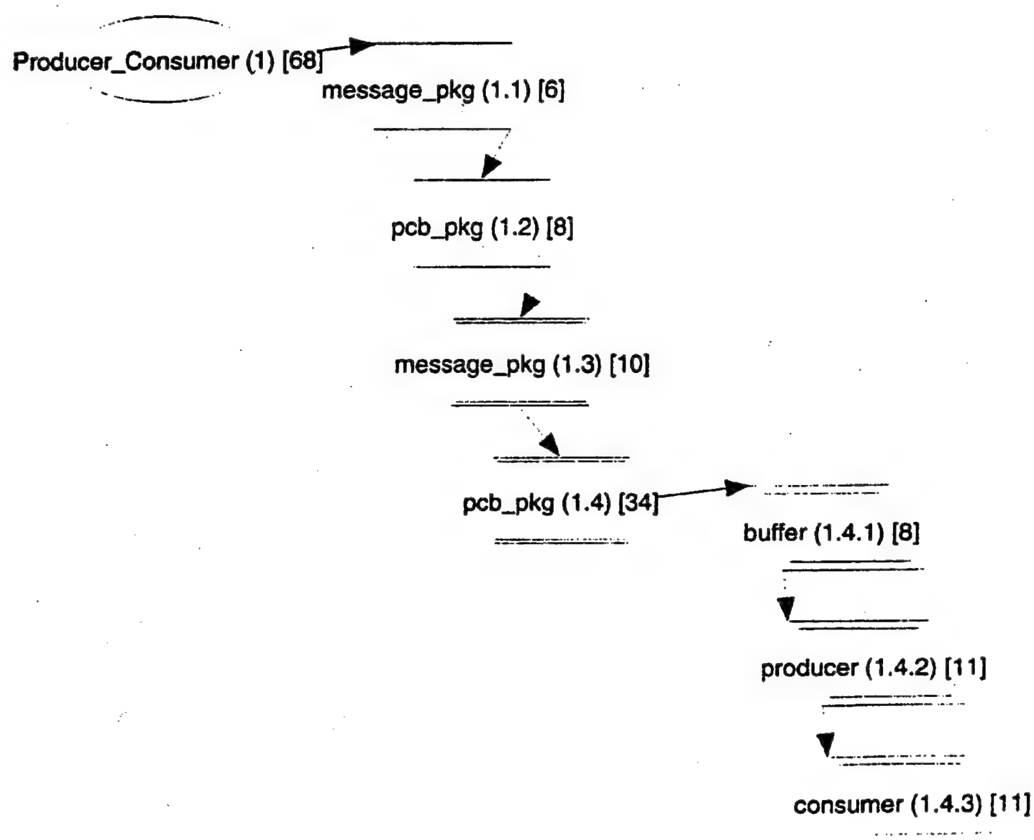
1. By constituents of block statements (pack., tasks, proc., etc.)
2. Bounded number of statements in Software Units (to facilitate graphic understanding)
3. SWU clustered based on "strength" of interfaces among them

Other rules for creating Software Units may be added.

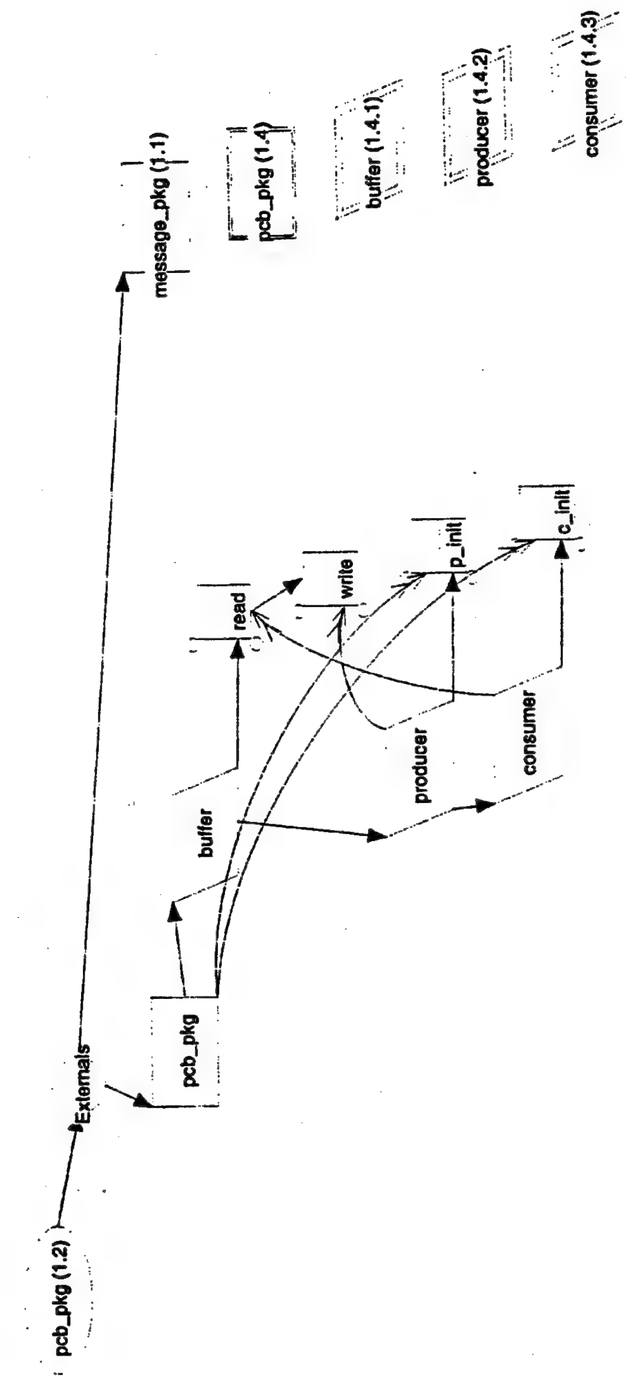
PCB_Pkg Scope Graph

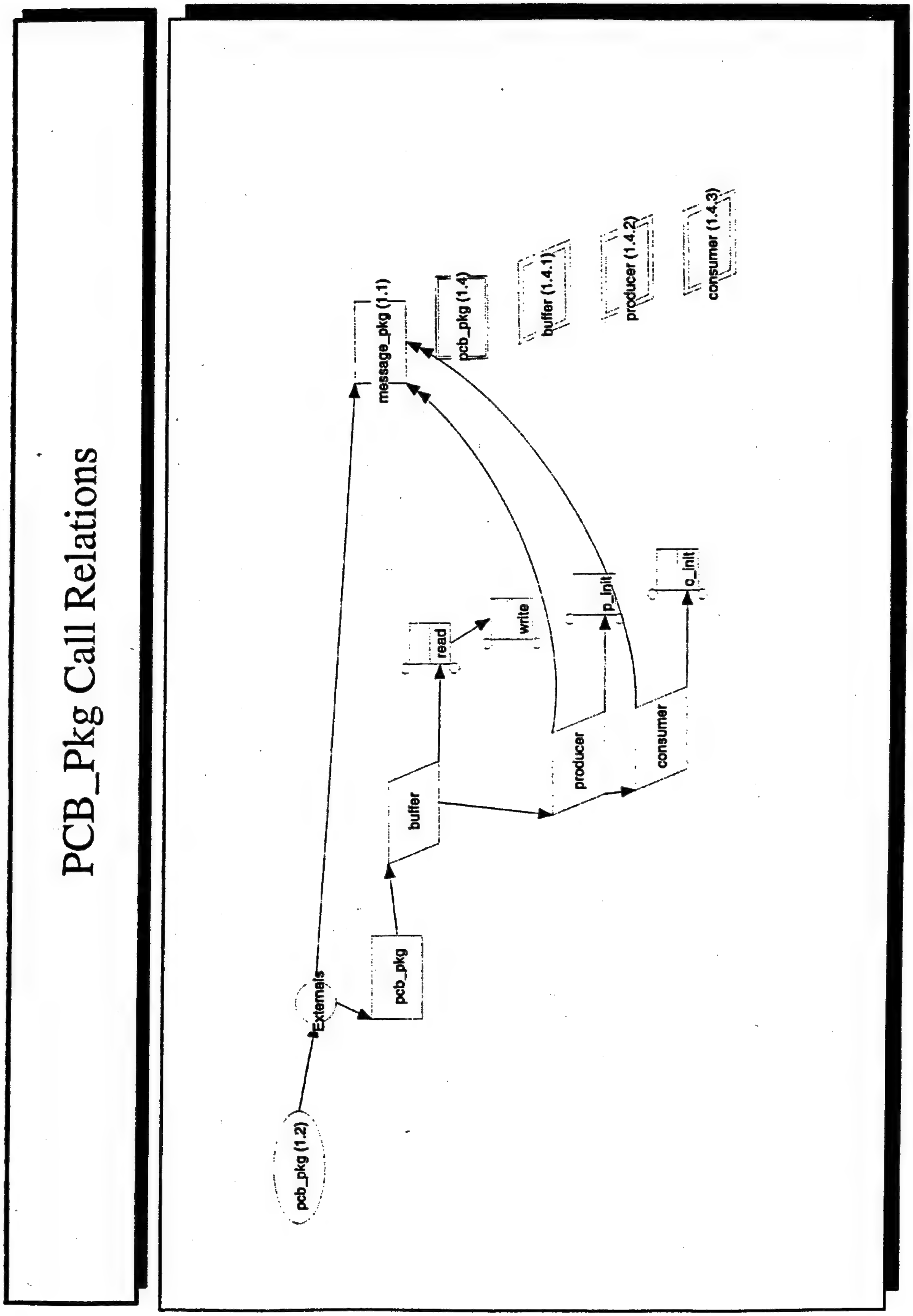


Producer_Consumer Unit Map

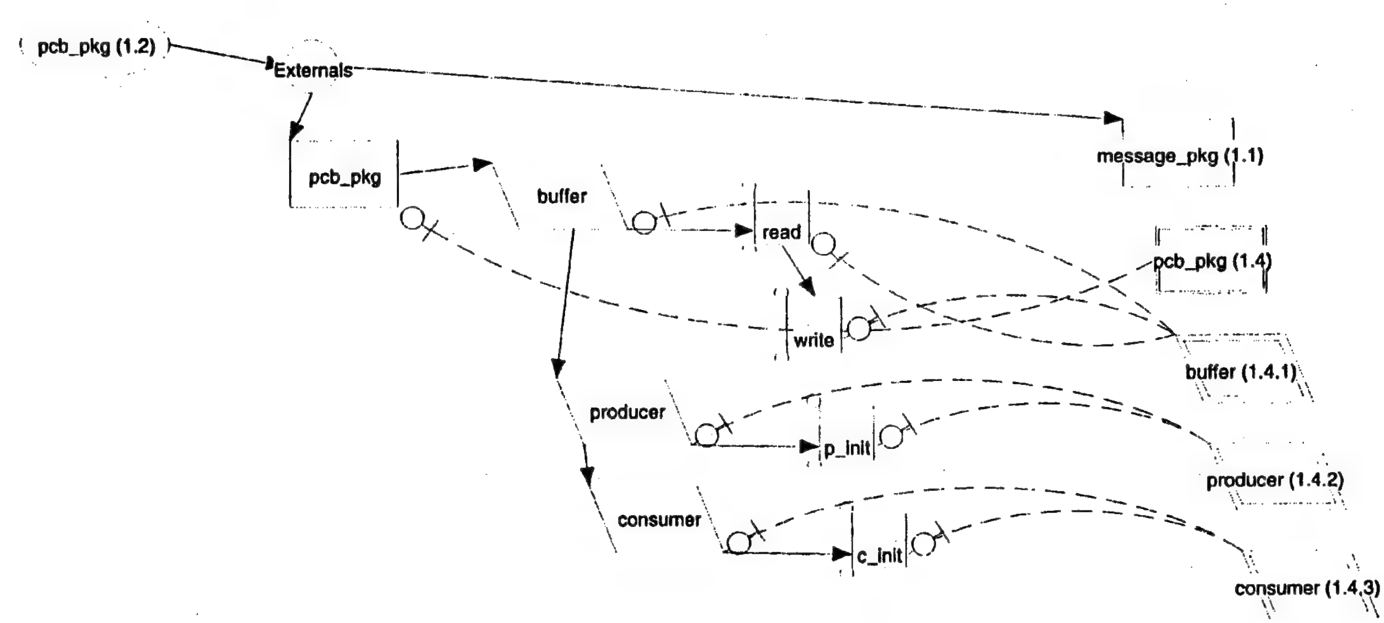


PCB_Pkg Message Relations

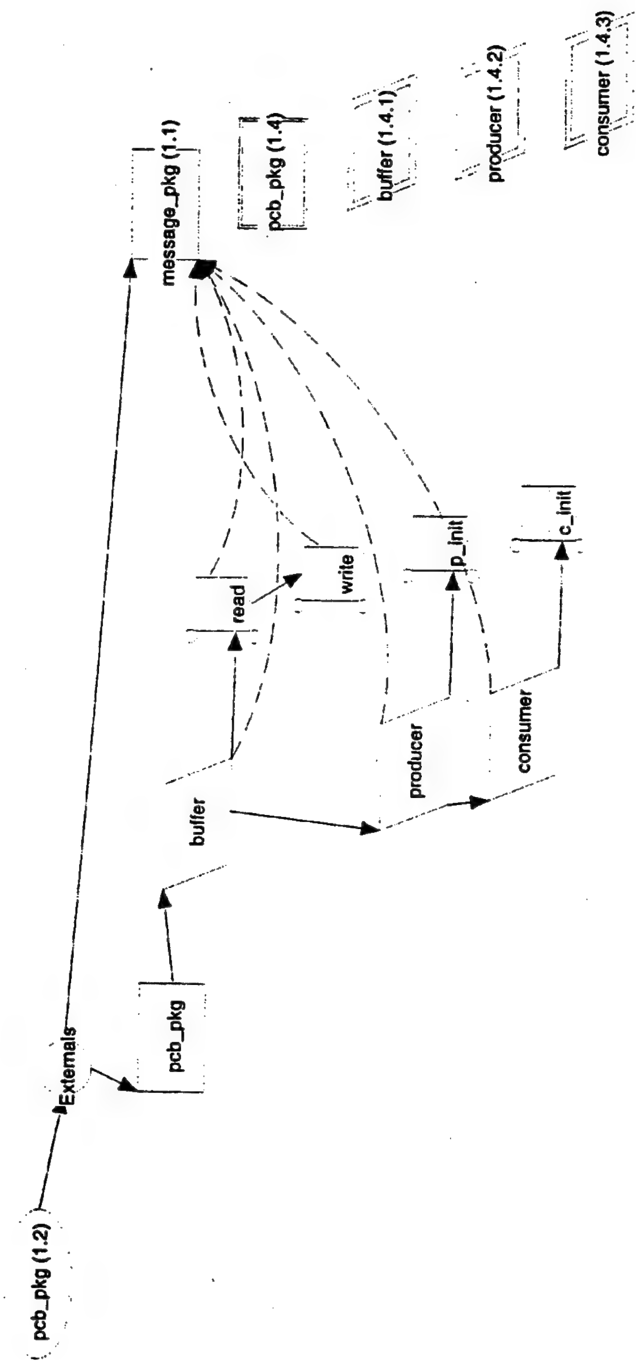



PCB_Pkg Call Relations
pcb_pkg (1.2)
Externals
pcb_pkg
buffer
read
write
producer
p_init
consumer
c_init
message_pkg (1.1)
pcb_pkg (1.4)
buffer (1.4.1)
producer (1.4.2)
consumer (1.4.3)

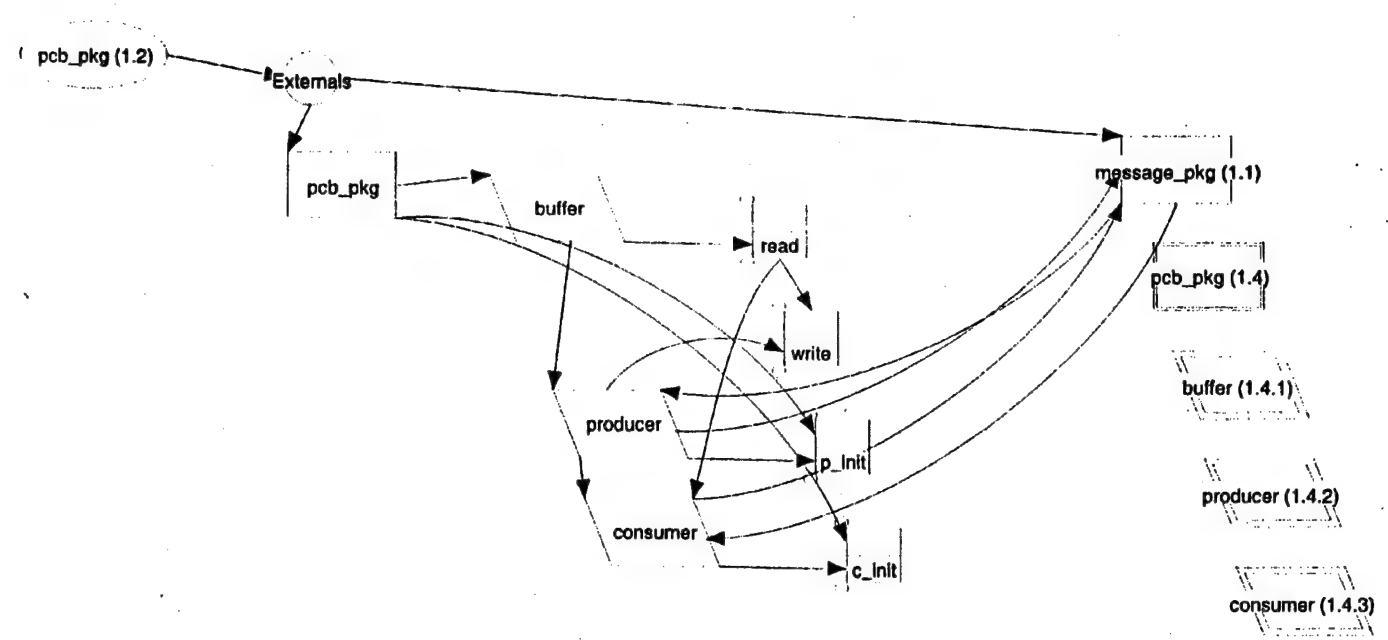
PCB_Pkg Context Relations



PCB_Pkg Type Relations

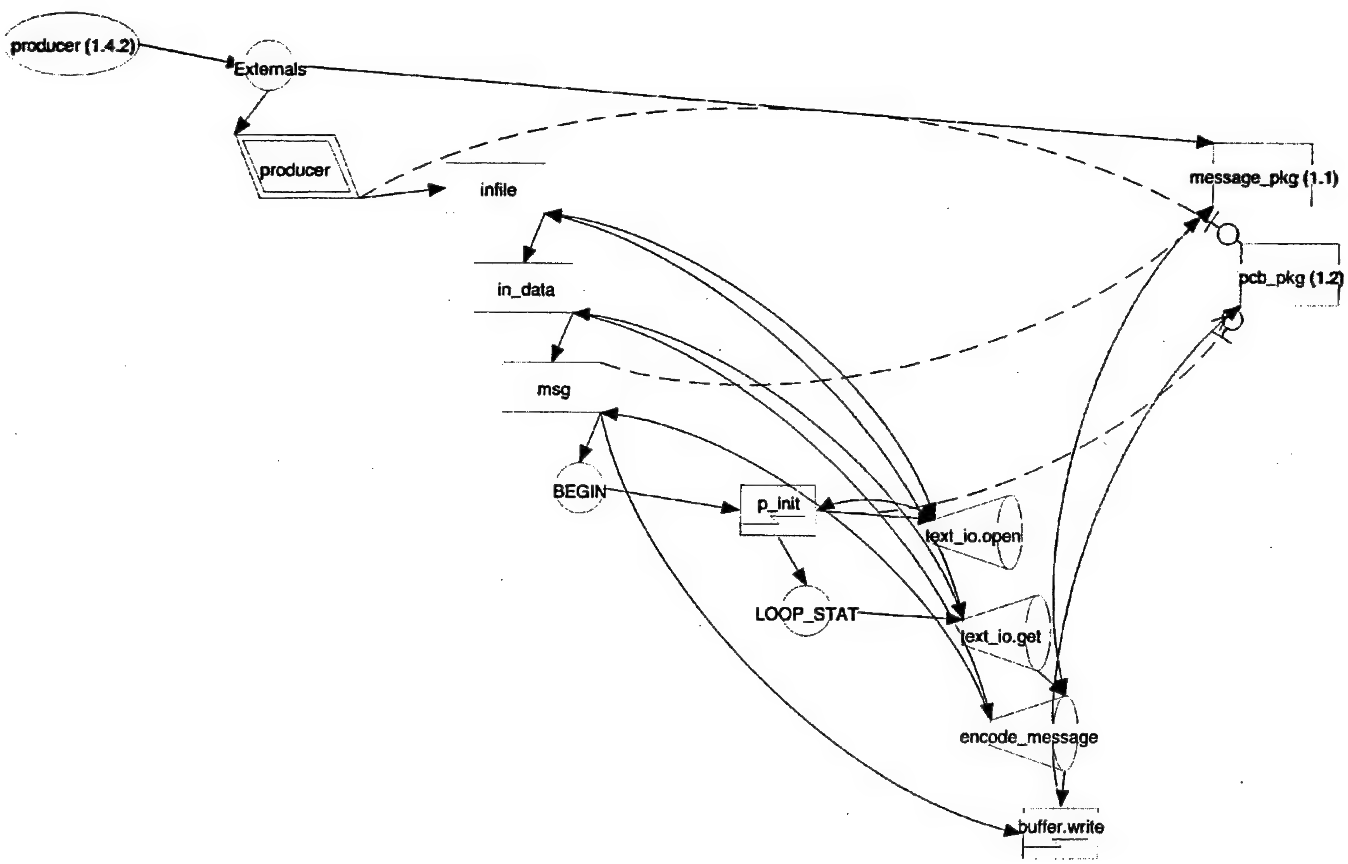


PCB_Pkg Memory Relations

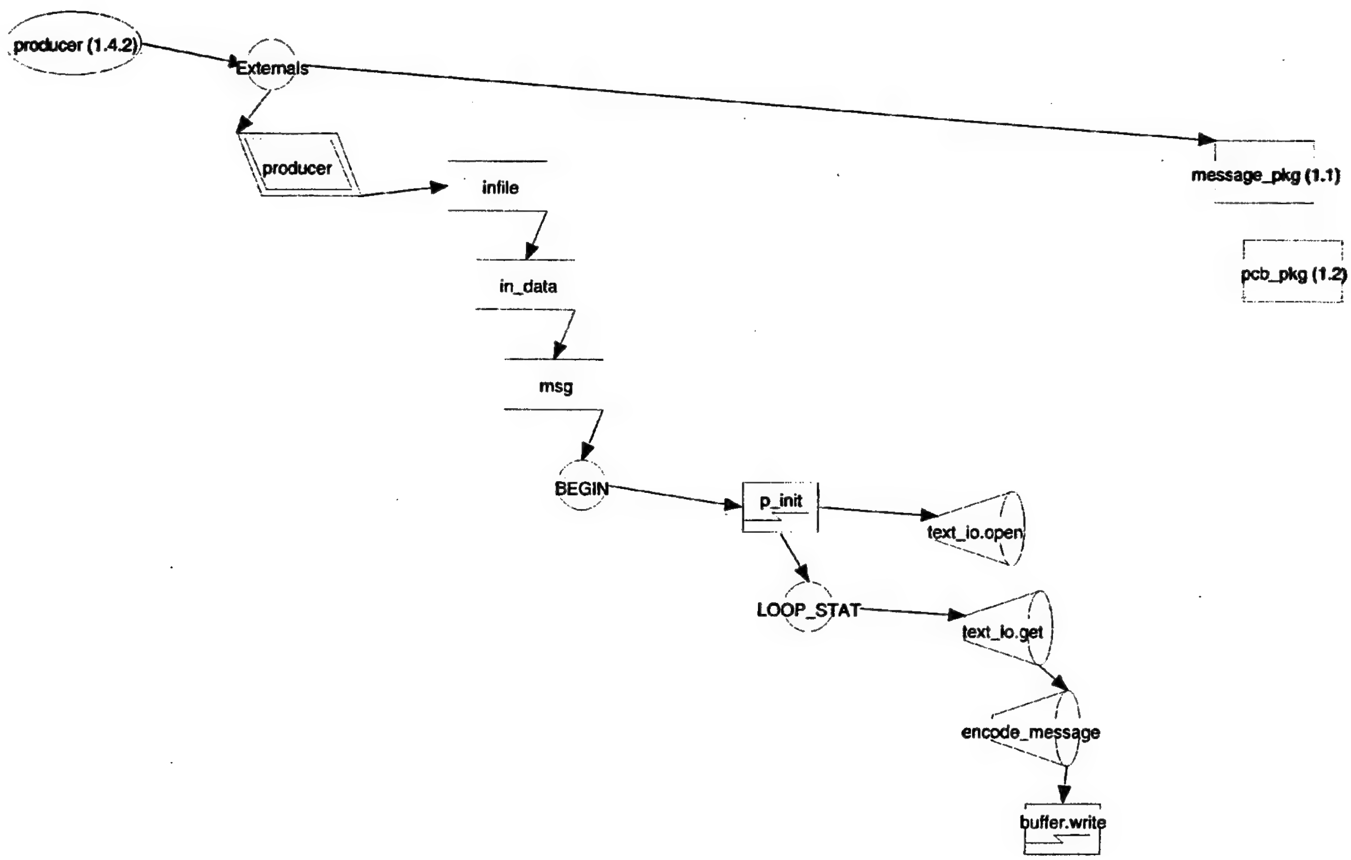


PCB_Pkg Interface Report

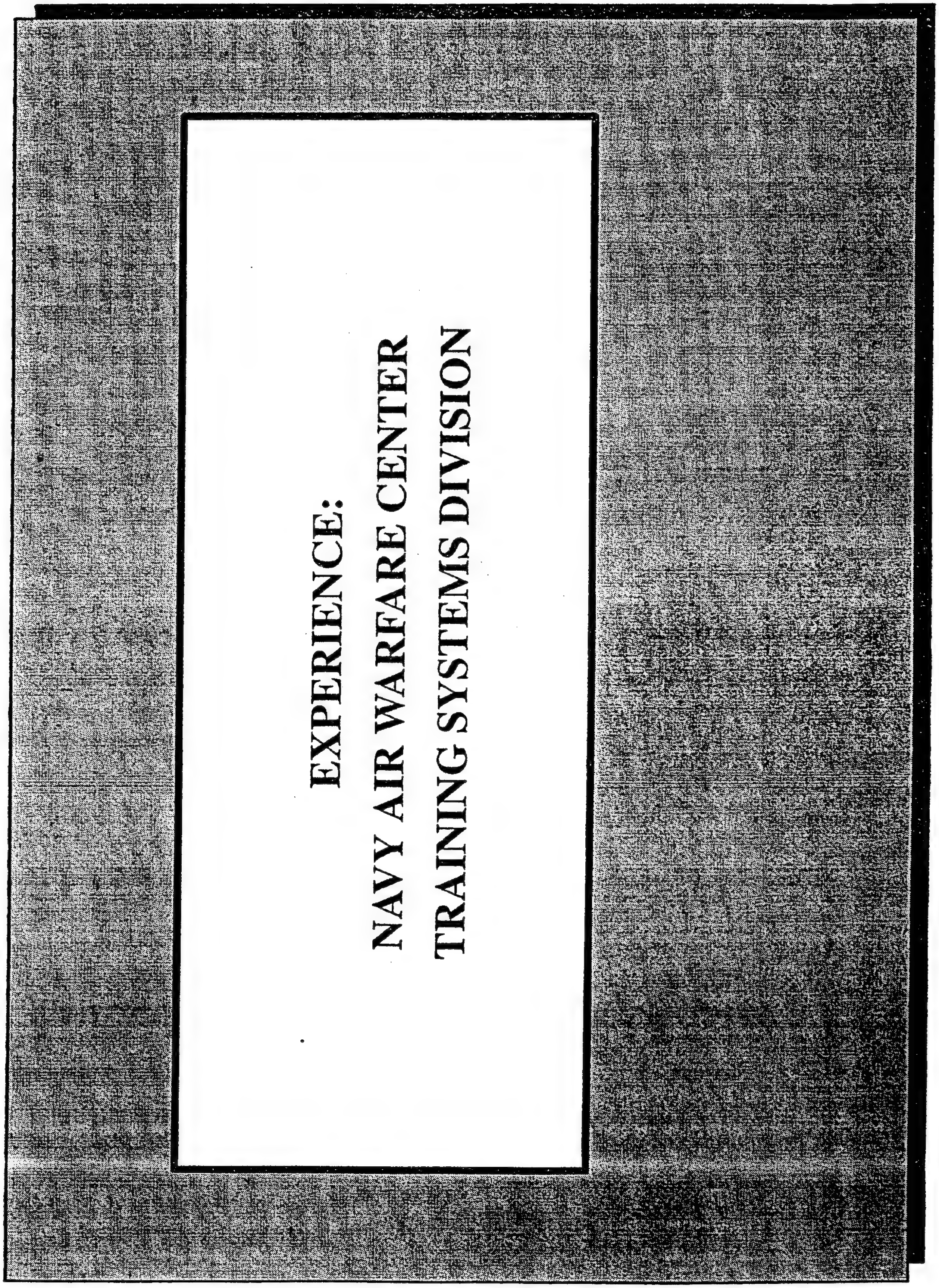
```
/*-------------------------------------------------------------------*/
/*                 INTERFACE REPORT for pcb_pkg (1.2)
/*-------------------------------------------------------------------*/
                   EXTERNAL                  INTERNAL
              -------------------      -------------------
CALL:
  1: [PROCEDURE encode_message ( data : IN integer ; msg : OUT message ) ; ] (1.1)
        <--- [encode_message ( in_data , msg ) ; ]
  2: [FUNCTION decode_message ( msg : IN message )  RETURN integer ; ] (1.1)
        <--- [out_data := decode_message(msg) ; ]


MEMORY:
  3: [PROCEDURE encode_message ( data : IN integer ; msg : OUT message ) ; ] (1.1)
        ---> [encode_message ( in_data , msg ) ; ]
  4: [PROCEDURE encode_message ( data : IN integer ; msg : OUT message ) ; ] (1.1)
        <--- [encode_message ( in_data , msg ) ; ]
  5: [FUNCTION decode_message ( msg : IN message )  RETURN integer ; ] (1.1)
        <--- [out_data := decode_message(msg) ; ]
  6: [FUNCTION decode_message ( msg : IN message )  RETURN integer ; ] (1.1)
        ---> [out_data := decode_message(msg) ; ]


TYPE:
  7: [TYPE message IS ] (1.1)
        <--- [msg : message ; ]
  8: [TYPE message IS ] (1.1)
        <--- [msg : message ; ]
  9: [TYPE message IS ] (1.1)
        <--- [ACCEPT read ( data : OUT message ) DO     ]
 10: [TYPE message IS ] (1.1)
        <--- [ACCEPT write ( data : IN message ) DO     ]
 11: [TYPE message IS ] (1.1)
        <--- [contents : message ; ]
 12: [TYPE message IS ] (1.1)
        <--- [ENTRY write ( data : IN message ) ; ]
 13: [TYPE message IS ] (1.1)
        <--- [ENTRY read ( data : OUT message ) ; ]


CONTEXT:
 14: [TASK BODY buffer IS ] (1.4.1)
        <--- [TASK buffer IS ]
 15: [ACCEPT read ( data : OUT message ) DO     ] (1.4.1)
        <--- [ENTRY read ( data : OUT message ) ; ]
 16: [ACCEPT write ( data : IN message ) DO     ] (1.4.1)
        <--- [ENTRY write ( data : IN message ) ; ]
 17: [TASK BODY producer IS ] (1.4.2)
        <--- [TASK producer IS ]
 18: [ACCEPT p_init ( filename : IN string ) DO     ] (1.4.2)
        <--- [ENTRY p_init ( filename : IN string ) ; ]
 19: [TASK BODY consumer IS ] (1.4.3)
        <--- [TASK consumer IS ]
 20: [ACCEPT c_init ( filename : IN string ) DO     ] (1.4.3)
        <--- [ENTRY c_init ( filename : IN string ) ; ]
 21: [PACKAGE BODY pcb_pkg IS ] (1.4)
        <--- [PACKAGE pcb_pkg IS ]
```



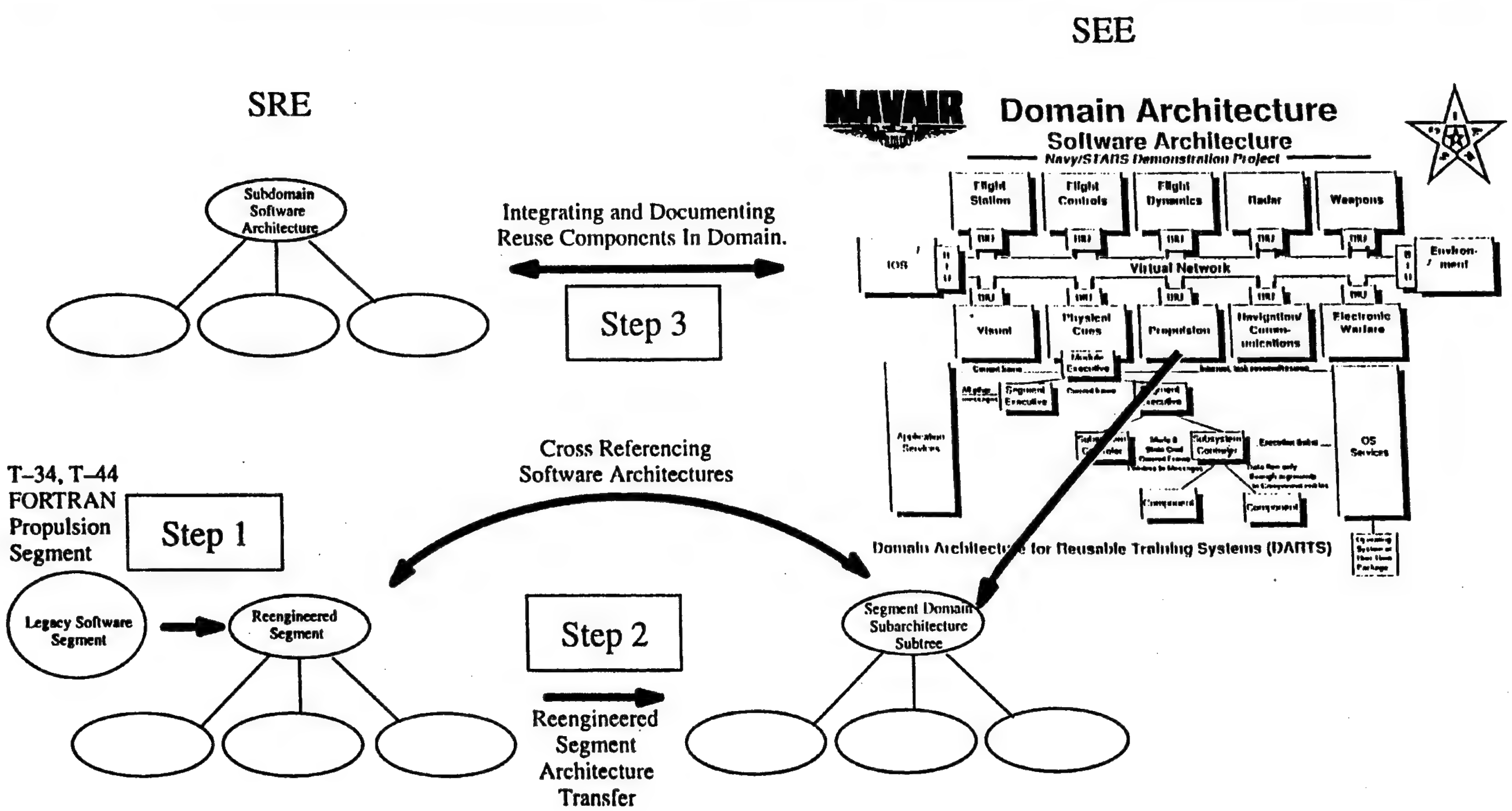
Producer (1.4.2) Relations Graph



Producer (1.4.2) Scope Graph


EXPERIENCE:
NAVY AIR WARFARE CENTER
TRAINING SYSTEMS DIVISION

Fitting Components Derived from Legacy Propulsion Segment of T–34, T–44 into High Level DARTS Software Architecture



Process Steps for Fusion AVTS Domain with Propulsion Components of Trainers of T–34 and T–44 Legacy Code

Step 1: Legacy code processing: Translated T–34 and T–44 FORTRAN propulsion code to Ada and produced documentation (a map, and for each software unit: code diagrams, data flow diagram, interfaces, comments and Ada code). Then tested the components code to assure that they are operating as expected using the T–34 test platform.

Step 2: Legacy code reengineering: Provided textual description of capabilities of the software units produced in Step 1. Matched these capabilities with the decision model for the propulsion segment of DARTS. The decision model shows common and variable requirements of components.

Step 3: Integration of reengineered legacy code into DARTS domain architecture: Modified interfaces of components to adhere to DARTS communications scheme, thus creating reusable components. An estimate of $15/LOC was made at the end of this step.

Step 4: Application engineering: This step is currently underway. It will create T–34 application software for propulsion segment from the DARTS reuse components developed in Step 3.

Transforming Software Architecture Requirements: Adapting Reuse Candidate Legacy Architectural Components for Inclusion in a Domain Architecture

1. Select from the legacy code a set of components with least interfacing to an external environment.

2. Generate a wrapper program for testing the reuse candidate component.

3. Generate a wrapper program for incorporating a reuse candidate component in the target domain architecture.

HCIL

Re-Engineering User Interfaces for the Maryland Department of Juvenile Justice

Anne Rose

Human-Computer Interaction Laboratory
University of Maryland
College Park, MD 20742
rose@cs.umd.edu

December 4, 1995

HCIL

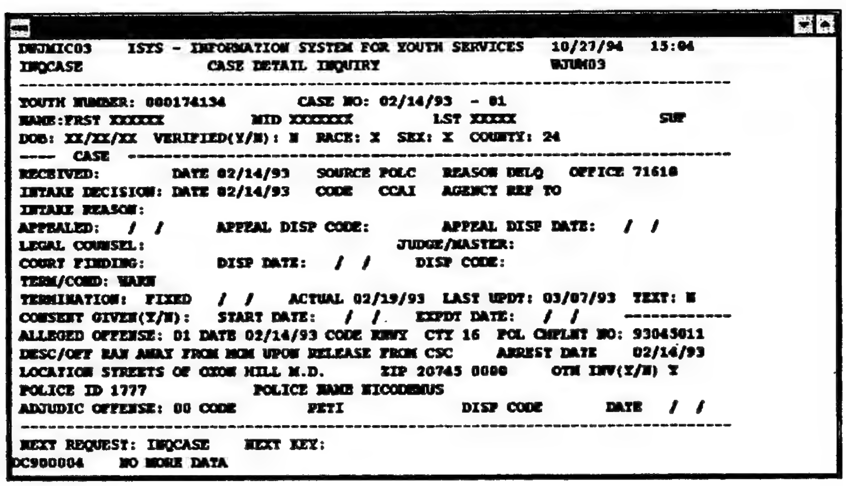
Our Goal

To make recommendations for developing an information system that effectively meets the needs of DJJ, with an emphasis on the user interface design

Introduction to ISYS

- Information System for Youth Services
- terminal based system
- used to process juvenile case referrals in Maryland
- 50,000 cases per year
- approximately 600 users

Sample ISYS Screen



```
DWJMIC03    ISYS - INFORMATION SYSTEM FOR YOUTH SERVICES     10/27/94    15:04
INQCASE                 CASE DETAIL INQUIRY                   WJUM03
--------------------------------------------------------------------------------
YOUTH NUMBER: 000174134         CASE NO: 02/14/93  - 01
NAME:FRST XXXXXX            MID XXXXXXX          LST XXXXX                 SUF
DOB: XX/XX/XX  VERIFIED(Y/N): N  RACE: X  SEX: X  COUNTY: 24
----  CASE  --------------------------------------------------------------------
RECEIVED:           DATE 02/14/93   SOURCE POLC   REASON DELQ   OFFICE 71610
INTAKE DECISION: DATE 02/14/93   CODE    CCAI    AGENCY REF TO
INTAKE REASON:
APPEALED:   / /       APPEAL DISP CODE:        APPEAL DISP DATE:   / /
LEGAL COUNSEL:                            JUDGE/MASTER:
COURT FINDING:        DISP DATE:   / /     DISP CODE:
TERM/COND: WARN
TERMINATION:  FIXED   / /    ACTUAL 02/19/93  LAST UPDT: 03/07/93  TEXT: N
CONSENT GIVEN(Y/N):   START DATE:   / /.   EXPDT DATE:   / /    -----------
ALLEGED OFFENSE: 01 DATE 02/14/93 CODE RNWY  CTY 16  POL CMPLNT NO: 93045011
DESC/OFF RAN AWAY FROM MOM UPON RELEASE FROM CSC     ARREST DATE    02/14/93
LOCATION STREETS OF OXON HILL M.D.       ZIP 20745 0000     OTH INV(Y/N) Y
POLICE ID 1777            POLICE NAME NICODEMUS
ADJUDIC OFFENSE: 00 CODE        PETI              DISP CODE      DATE   / /
--------------------------------------------------------------------------------
NEXT REQUEST: INQCASE    NEXT KEY:
DC900004    NO MORE DATA
```

Steps to Achieve Goal

- Evaluate ISYS and assess user needs.
- Recommend improvements to existing system.
- Propose designs for the next generation ISYS.
- Recommend a software methodology for implementing the new system.

ISYS Evaluation

- Process
  - read documentation
  - 22 visits to several DJJ offices
  - administered QUIS
  - hands-on experience with ISYS
- Papers
  - An Applied Ethnographic Method for Redesigning User Interfaces
  - User Interface Reengineering: A Diagnostic Approach
  - Assessing User's Subjective Satisfaction with the Information System for Youth Services (ISYS)

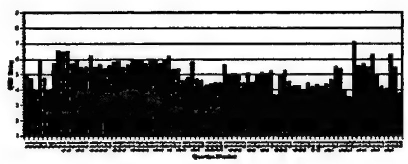
HCIL Benefits of Ethnographic Evaluation

- learned how system was really being used
- humanized user problems
- increased trustworthiness and credibility
- users became increasingly active participants in the design process

HCIL Questionnaire for User Interaction Satisfaction (QUIS)

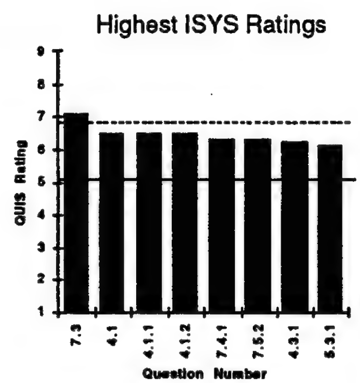
- developed by HCIL, proven reliability and validity
- customized to assess ISYS
- administered to 332 DJJ personnel

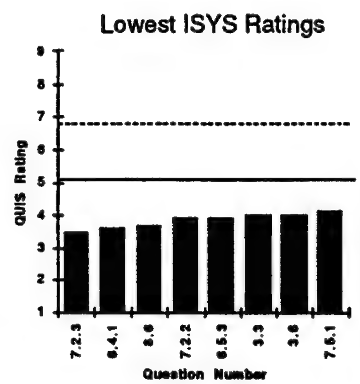
ISYS Mean Ratings



The overall mean (5.1) indicates that ISYS rates below average compared to other systems rated by the QUIS.

Highest and Lowest ISYS Ratings





HCIL QUIS Comments

- Compiled electronic database of comments
- Categorized according to type
- Comments ranged from "very frustrated" to "no problems"

HCIL Short Term Recommendations

- improvements to existing system
- 28 recommendations
  - system access
  - data display
  - data entry
  - consistency
  - error messages
  - functionality
- estimated payoff vs. effort

Improving the Login Procedure

ISYS Login Procedure:

Type:	BDCDEV
Press:	Clear key
Type:	CSSN
Type:	*login id*
Type:	*password 1*
Type:	DBDC
Type:	*login id*
Type:	*password 2*
Select:	ENTC menu option
Type:	WJM
Type:	*office code*

Suggested Procedure:

Type:	*login id*
Type:	*password*

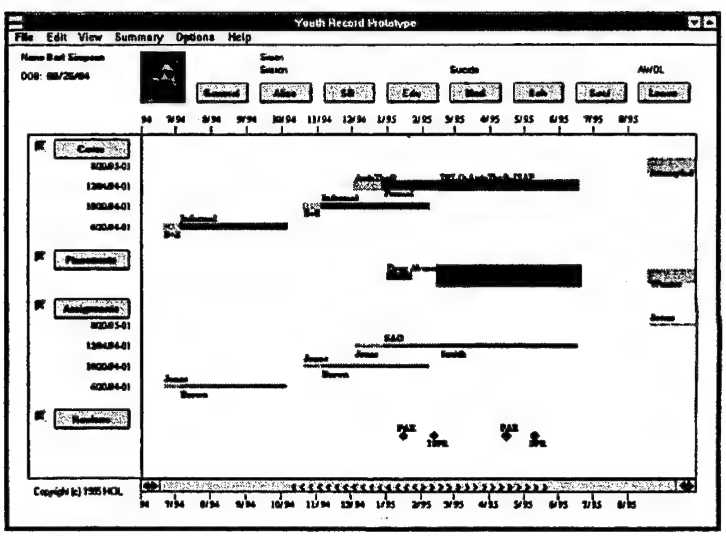
Prototypes

- address needs discovered during evaluation
- possible add-ons to existing system
- 3 prototypes
  - DJJ Navigator (help manage workload)
  - LifeLines (present youth record in single screen)
  - IVEE (visualize aggregate information and explore trends)

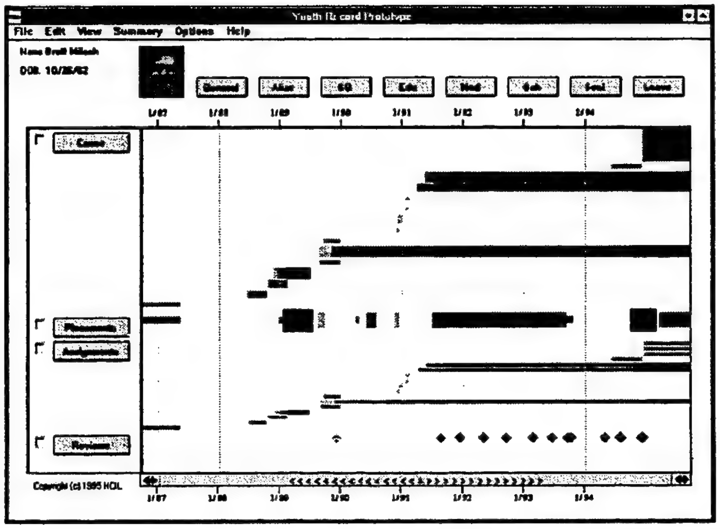
LifeLines Overview

- Single screen overview of all cases, placements, assignments and reviews associated with a youth
- Direct access to frequently used ISYS screens
- Zoom on smaller time period
- Highlight relationships
- Potential add-on to current ISYS for PC users
- "Life-Line" addresses the generic problem of providing overview of a person's life (e.g. medical record, resume)

Simple Youth Record



Complex Youth Record



User's Feedback

- Visual Basic prototype
- 60 users (20 minute demo + try + questions)
- Most users enthusiastic
- A few worried about use of color and thickness
- Major pluses: overview + quick access to details
- Many alternate layouts proposed (control panels?)

LifeLines Issues

- Policy for use of color and thickness
- Optimization of the layout/labeling
- Test readability/usability with real users
- Data entry (an "Add menu" vs. editable timelines)

To know more...

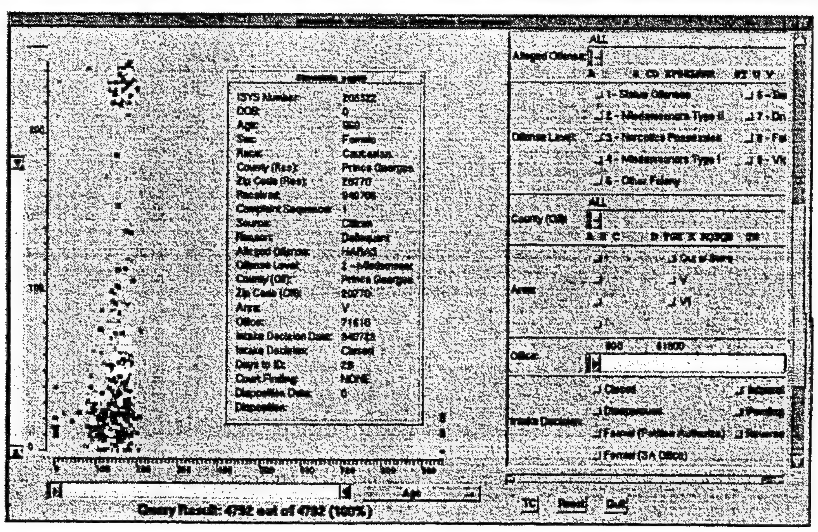
- Plaisant, C., Milash, B., Rose, A., Widoff, S., Shneiderman, B., LifeLines: Visualizing Personal Histories, to appear in Proc. of CHI 96, ACM, New York.

- Video available in HCIL VideoReport 95 and a revised version will appear in the CHI 96 video.

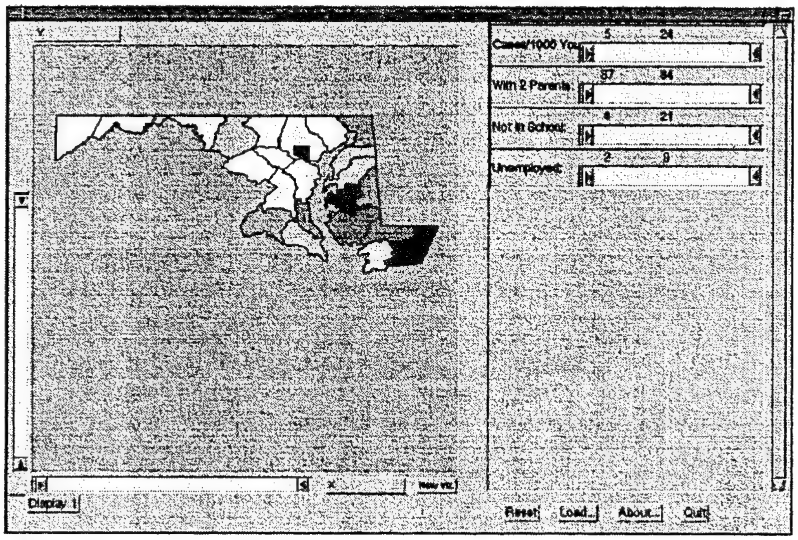
IVEE Overview

- Information Visualization & Exploration Environment (IVEE)
- Research tool being developed by Chris Ahlberg and Erik Wistrand of Chalmers University, Sweden
- Dynamic exploration of data trends using zooming and filtering
- Supports generic datasets
- Possible export of subsets to other applications

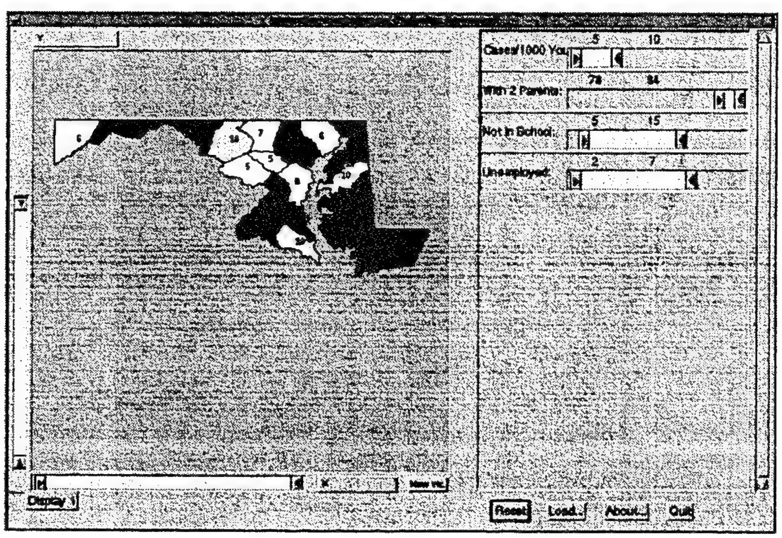
July 1994 Intake Cases



Influences on Delinquency



Influence of Two Parent Families



HCIL IVEE Issues

- Dealing with large datasets: speed and clutter
  - Doan, K., Plaisant, C., Shneiderman, B., Query Previews in Networked Information Systems, HCIL CS-TR-3524, University of Maryland, College Park, MD, 1995.
- Better date handling
- Other test datasets
- Collect feedback from potential users

HCIL To know more ...

- Ahlberg, C., Wistrand, E., IVEE: An Information Visualization & Exploration Environment, Proc. of IEEE Visualization 95.
- Video available in HCIL VideoReport 95

HCIL Current Status

- Working with Cognetics Corp. to prepare RFP
- Testing Cognetics Design Methodology (CDM)
- Report generation review
- Soon will work on overall prototype

1

Data Reengineering

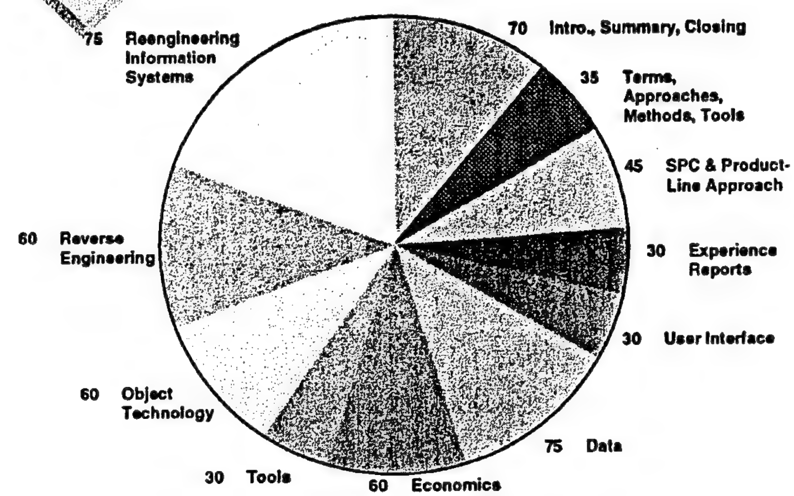
contribution to the
2nd SPC REENGINEERING WORKSHOP:
Approaches to Reengineering for Information Systems

Software Productivity Consortium
Herndon, Virginia December 4 and 5, 1995

Peter Aiken, Ph.D. - Virginia Commonwealth University
paiken@cabell.vcu.edu - 804/828-0174

2

Time Allocation



3

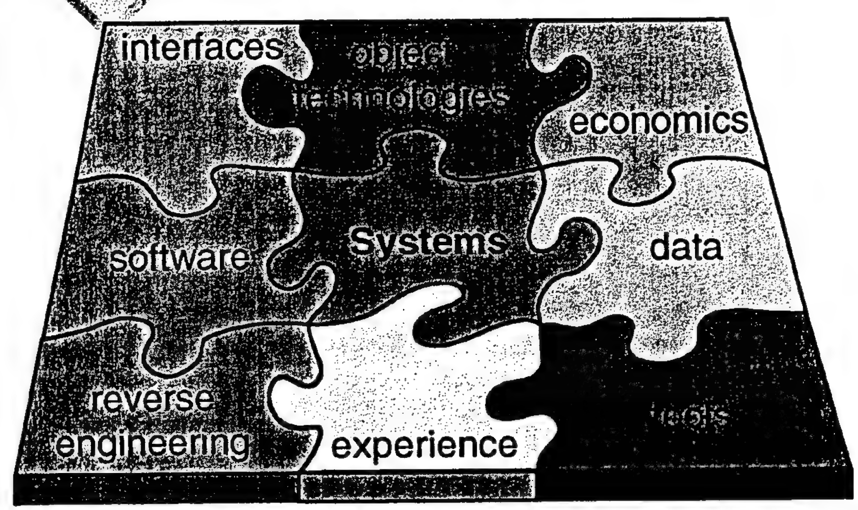
What can be reengineered?

from the program

- Information systems
- User interfaces
- Software
- Data

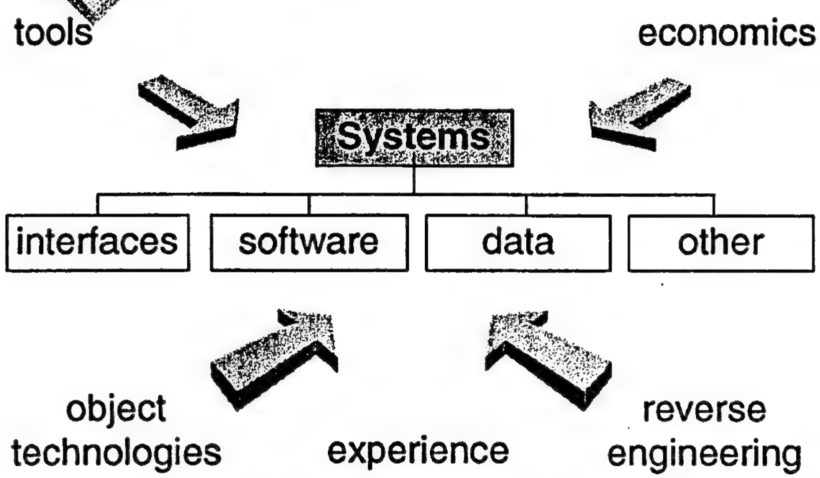
4

How do the pieces fit together?



5

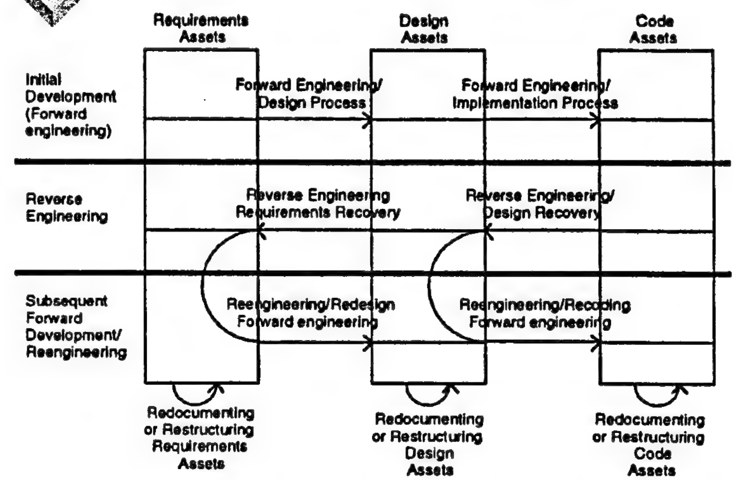
Topical Map



6

(adapted from Chikofsky and Cross "Reverse Engineering and Design Recovery: A Taxonomy" © 1990 IEEE Software)

What can we learn about reengineering?



7

Definition of Data

At least one *fact* paired with at least one *meaning*

fact	fact	meaning	meaning
fact	fact	meaning	meaning
fact	fact	meaning	meaning
fact	fact	meaning	meaning
fact	fact	meaning	meaning
fact	fact	meaning	meaning

Organizational Data Bank

Appleton [1984]

8

Definition of Data

Each unique combination of facts and meanings defines an individual data

fact	fact	meaning	meaning
230	fact	meaning	meaning
fact	fact	meaning	meaning
fact	fact	meaning	meaning
fact	fact	period sales	meaning
fact	fact	meaning	meaning

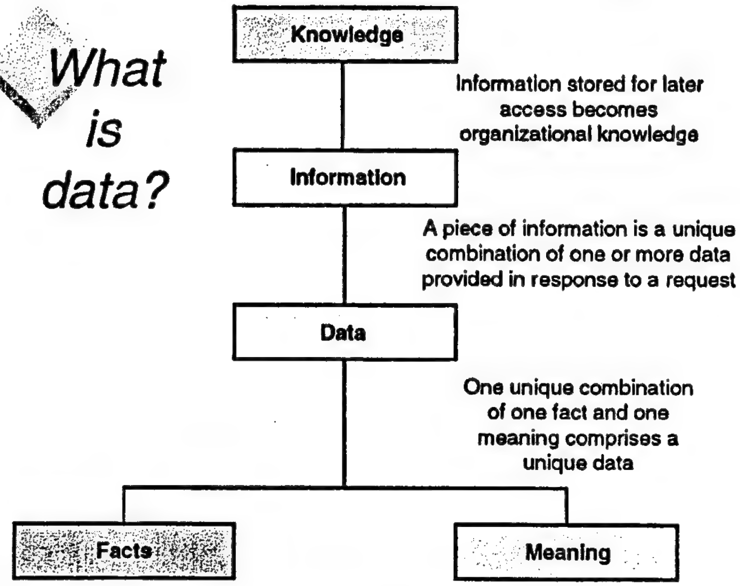
Organizational Data Bank

Appleton [1984]

9

What is data?

(Facts, Meaning, Data, and Information Adapted from Appleton 1982)



10

Definition of Information

When data is supplied in response to a request, it becomes information

How many fewer units are in the warehouse this period? service manager query

fact	fact	meaning	meaning
230	fact	meaning	meaning
fact	fact	meaning	*inventory reduction*
fact	fact	meaning	meaning
fact	fact	*period sales*	meaning
fact	fact	meaning	meaning

Organizational Data Bank

sales manager query

How many units sold for the period?

Appleton [1984]

11

Definition of Information

Different facts and meanings can be combined into data and supplied as information in response to different queries

How many fewer units are in the warehouse this period? service manager query

fact	fact	meaning	meaning
230	fact	meaning	meaning
fact	fact	meaning	*inventory reduction*
fact	fact	meaning	meaning
fact	fact	*period sales*	meaning
fact	fact	meaning	meaning

Organizational Data Bank

sales manager query

How many units sold for the period?

Appleton [1984]

12

Data Reengineering Goals

1. difficult to forecast future information needs - stored information needs to be flexible and adaptable
2. data independence is a goal
3. leverage economies of scale by managing lot of information with a relatively small amount of data

13

Process/Data Independence

- Tight coupling between organizational processes and data makes it awkward to maintain and change either
- Changes to either processing or data require corresponding and often extensive modifications to the other
- Situation brittleness can be eliminated by separating process and data with the development of an information architecture

14

Legacy Information Systems

- Intriguing role
- Chief obstacle to enterprise integration
- Simultaneously, chief enabler of enterprise integration
- Valuable sources of information
- Means of leveraging the existing information system investment

15

Sharable Data

- Major enabler as well as an indicator of organizational dexterity
- Without sharable data - more resources are required to produce needed information
- A prerequisite to effective use of enterprise information as a strategic asset and development of an information architecture

16

Dexterous Organizations ...

- More capable of responding effectively to environmental opportunities
- Can quantitatively evaluate processes by measuring output production
- Time required to introduce a new product to the market
- Resources required to process the accounts payable

17

Dexterous Organizations ...

- ❖ Knowledge of capabilities
- ❖ Useless if not applied in appropriate contexts with reasonable expectations
- ❖ Example: facility of major airline reservation systems to handle:
  - – tens of millions of monthly transactions
  - – two thousand messages per second
  - – 500,000 new passenger name records daily [Hopper 1990]

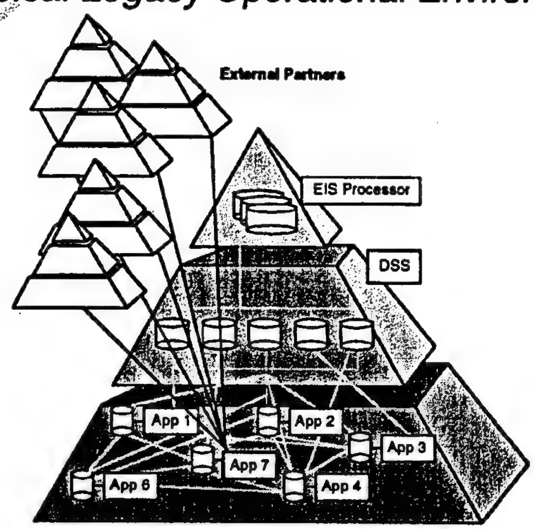
18

Shared Data

- ❖ Enterprise integration is impossible without integrated enterprise information
- ❖ Shared data is typified by organizational ability to use information as a strategic asset
- ❖ However, assets are useless without knowledge of the asset characteristics

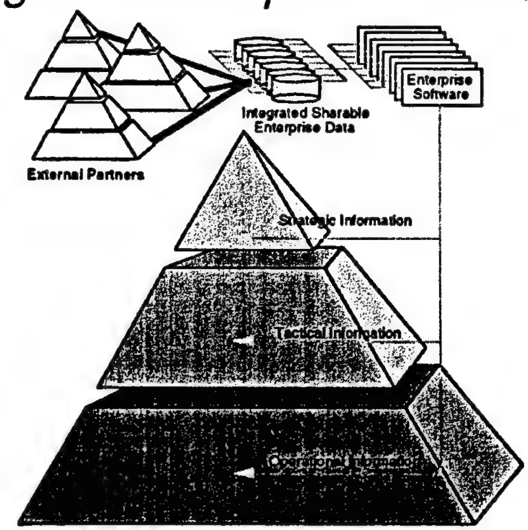
19

Typical Legacy Operational Environment



20

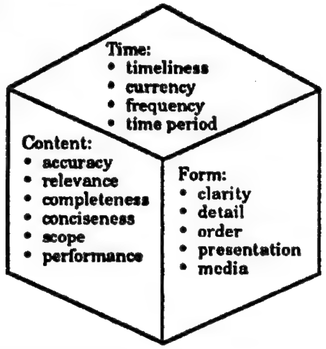
Integrated Enterprise Information



21

Attributes of Information Quality

(adapted from [O'Brien 1993])



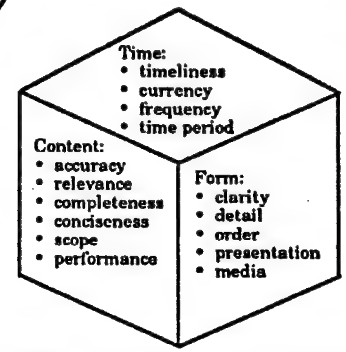
Time Dimension

◆ Timeliness	provided when it is needed
◆ Currency	up-to-date when it is provided
◆ Frequency	provided as often as needed
◆ Time Period	be provided about past, present, and future time periods

22

Attributes of Information Quality

(adapted from [O'Brien 1993])



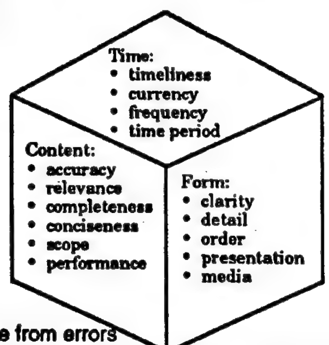
Form Dimension

◆ Clarity	provided in a form that is easy to understand
◆ Detail	provided in detail or summary form
◆ Order	arranged in a predetermined sequence
◆ Presentation	presented in narrative, numeric graphic or other forms
◆ Media	provided on printed paper documents, video displays, or etc.

23

Attributes of Information Quality

(adapted from [O'Brien 1993])

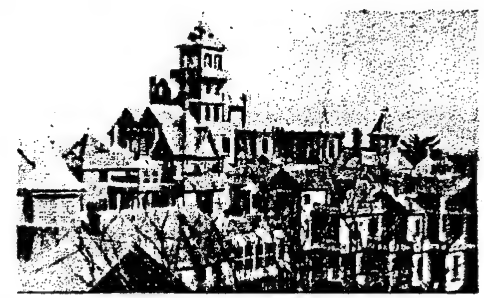


Content Dimension

◆ Accuracy	free from errors
◆ Relevance	related to situation specific recipient information needs
◆ Completeness	all the information that is needed should be provided
◆ Conciseness	only the information that is needed should be provided
◆ Scope	broad/narrow, or internal/focus
◆ Performance	measured to reveal performance

24

Winchester House



- compares information systems portfolios to the Winchester House -- a building where construction
- "continued 24 hours a day, seven days a week, month after month, year after year–for 38 years! ... highlights of the tour (now given) are such odd features as stairways that rise into ceilings, doors and windows blocked by walls, more passageways and halls than rooms, a four story chimney that falls short of the roof, and many rooms serving the same purpose."

Enterprise Architecture Planning [Spewack 1993]

25

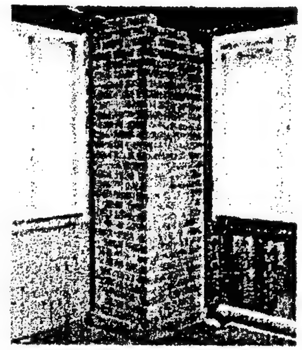
Winchester House (example)

In this example, a person walks down 7 steps and up 11, gaining only 4 steps but apparently satisfying a mystical need for Sarah Winchester

Winchester Mystery House [Roberts]

26

Winchester House (example)



This strange chimney begins on the ground floor and goes up through 4 floors ... only to stop just inches short of the 4th story roof ... making the 3 or 4 fireplaces that connect to it absolutely useless.

Winchester Mystery House [Roberts]

27

Winchester House

- No overall set of blueprints showed what Mrs. Winchester wanted her house to be.
- Similarly, most systems organizations have no overall blueprints for the data, systems and technology needed to support the business.

Enterprise Architecture Planning [Spewack 1993]

28

National Cathedral Development

- spanned eighty years and four generations of craftsman - 'Could it have been done without the blueprints?'
- Similarly, after the passage of time - information system:
- "plans and documentation ... are poor or non-existent, and the original designers and craftsmen are no longer available. The result is increasing maintenance costs and decreasing programmer productivity-a situation that is inefficient, wasteful and costly to our businesses."

[Connall & Burns 1993]

29

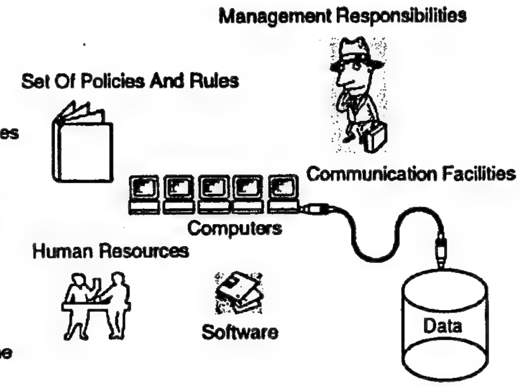
Architectures Defined

- Architectures are plans, guiding the transformation of strategic organizational information needs into specific information systems development projects. All organizations have architectures - some are better understood and documented than others.

30

Corporate Data Architecture Defined

- How and why do the components interact?
- Where do they go?
- When are they needed?
- Why and how will the changes be implemented?
- What should be managed organization-wide and what should be managed locally?
- What standards should be adopted?
- What vendors should be chosen?
- What rules should govern the decisions?
- What policies should guide the process?



(Adapted from [Allen & Boynton 1991])

31

Data Architectures Defined

- Blueprints/master plans for accomplishing data administration goals and objectives
- Information maintained by data administration in a data architecture includes the:
- sources and uses of data;
- creation and use of data by specific processes; and
- various organizational communication capabilities for delivering the information among data collections and its uses.

32

Zachman Framework

- Describes data architectures as integrated information collections pertaining to information systems development
- Key is to organize information according to different perspectives
- Views range from contextual to implementation specific data, function, and communication specifications

33

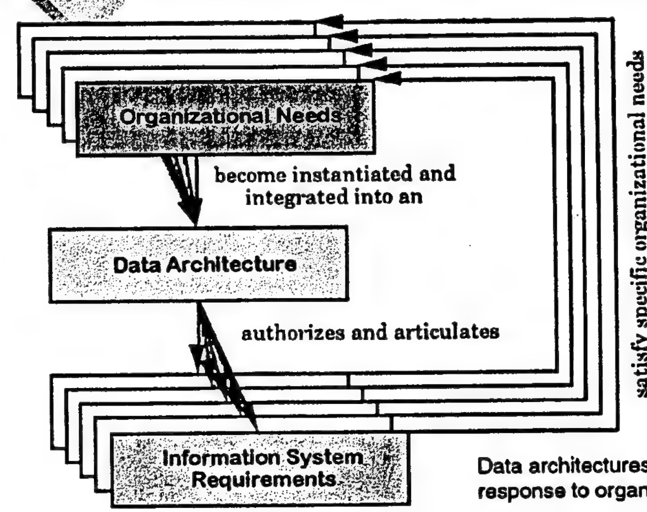
Zachman Framework

	(Nouns)	DATA (What?) (Verbs)	FUNCTION (How?)	NETWORK (Where?)
	Model components	Entities & Relations	Processes, Inputs/Outputs	Nodes & Lines
Investment Decisions	The ballpark view is concernedwith *organizational scope*and *objectives*	List of important business things Entity = class of business things	List of processes the business performs Process = class of business functions	List of business locations Node = business location
Business Entities	The owner view is concerned with modeling the *business units* comprising the organization	E.G., entity-relationship diagram Entity = business entity Reln = business rule	E.G., functional flow diagram Proc = bus. process I/O = bus. resources	E.G., logistics network Node = business unit Link = business relationship
Business Requirements	The analyst view is concerned with modeling the *information system specifications*	E.G., data model Entity = data entity Reln = data relationship	E.G., data flow diagram Proc = appl. function I/O = user views	E.G., distributed system architecture Node = IS function Line = line char.
System Requirements	The builder view is concerned with transforming the planner's *plan* into reality	E.G., data design Entity = segment/row Reln = pointer/key	E.G., structure chart Proc = comptr function I/O = screen/device fmt	E.G., system architecture Node = hard/software Line = line specification
Software Requirements	The maintainer view is concerned with understanding the *detailed representations of the system operation*	E.G., database definition Entity = fields Reln = addresses	E.G., Program Proc = language stmts I/O = control blocks	E.G., Network Architecture Node = addresses Link = protocols
	The actual system	data	function	communications

(Zachman 1987)

34

Data Architecture Development



Data architectures are developed in response to organizational needs

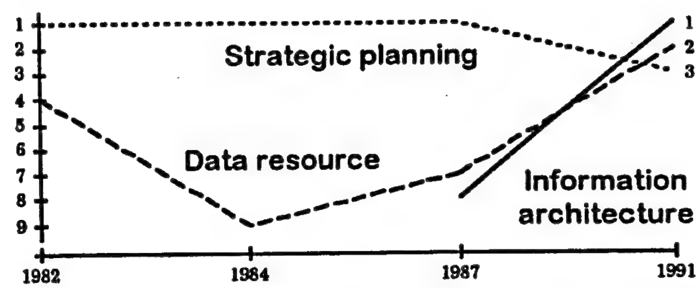
35

Data Architecture Development

- The successful development of a organizational data architecture requires a degree of information system development:
- more often spoken about than sought after and
- more often sought after than achieved.
- Strategic planning without the benefit of a data architecture is just a ritual rain dance.

36

Organizational Data Administration



Relative Importance of Data, Architecture and Planning Issues Identified by Information Systems Professionals (adapted from [Neiderman et. al 1991)

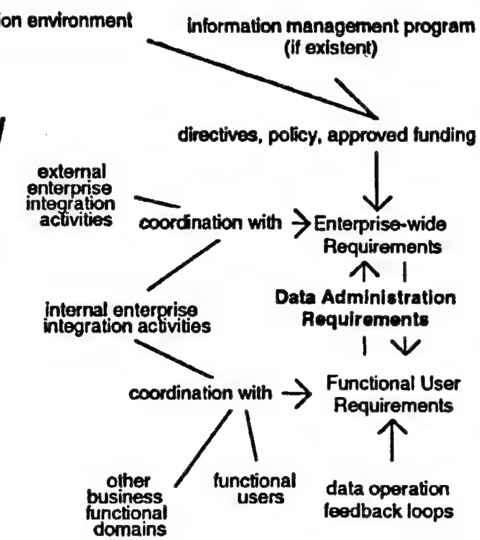
37

Organizational Data Administration

- developing and maintaining standard data products and models;
- developing and maintaining an organizational data bank for storing and integrating organizational data assets;
- encouraging the use of common procedures and tools; and
- providing education, training, and consultation services to the organization

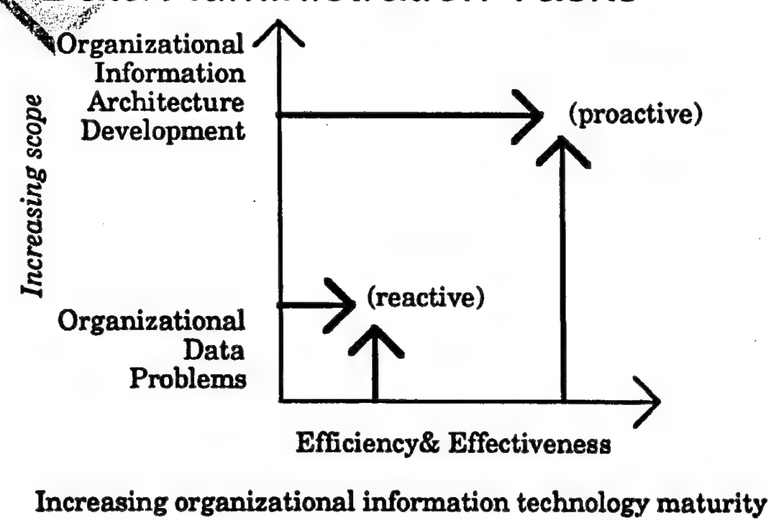
38

Organizational Data Administration



39

Data Administration Tasks



40

4 Types of Reengineering

- Data Reengineering
- Existing data are inventoried
- Structured into an architecture, and evolved into more flexible and process independent support for business processes

41

4 Types of Reengineering

Business Process Reengineering

- Inventories current supported business processes
- Corrects locally optimized process and
- Focus them on organization objectives

42

4 Types of Reengineering

Software Reengineering

- Reengineer selected software applications
- To obtain targeted software assets
- To obtain design assets for reuse
- Some for both reasons

43

4 Types of Reengineering

Infrastructure Reengineering

- Evaluate technological infrastructure for future opportunities
- Matching organizational needs with solutions facilitating organizational dexterity
- Solving problems such as removing barriers to inter-operability

44

Procedure Overview

Phase

1 What do you currently have?
   Reverse engineer existing assets by "as is" modeling

2 What would you like to have?
   Develop an architectural plan by modeling "to be" systems

3 What do you need?
   Identify the gaps

4 How do will you get there?
   Develop solutions to close the gaps and implement

45

Enterprise Integration Defined

- 'effective and efficient organizational functioning'
- improving the overall performance of large, complex systems
- processing efficiency, unit responsiveness, perceived quality, and product differentiation
- facilitating the interaction among organizations, individuals, and systems
- Proceedings of the First International Enterprise Integration Modeling Conference, 1992

46

Enterprise Integration Defined

- Enterprise integration efforts
  - facilitate the interaction among organizations, individuals, and systems
- Enterprise integration can be defined as
  - a state of organizational dexterity
  - combined with organizational awareness of that dexterity

47

Integration Perspectives

	Data	Process	Software	Infrastructure
"As-is" analysis (establish base line)	What data assets do we currently have?	What processes are we currently supporting?	What software applications are we currently supporting	What is our technological infrastructure base?
"To-be" analysis (radical change)	What data assets should we be maintaining?	What processes should be supported?	How should our software assets be employed?	What technological infrastructure will be required in the future?
Continuous implementation activities	How will we get from our current state to our desired state?	How are we going to implement the new processes?	Transforming our software assets to the desired states?	How are we going to implement the required infra-structure?

48

Integration Perspectives

	Data	Process	Software	Infrastructure
"As-is" analysis base line)	What data assets do we currently have?	What processes are we currently supporting? structure?	What software applications are we currently supporting	What is our technological infrastructure base?
analysis (radical change)	What data assets should we be maintaining?	What processes should be supported?	How should our software assets be employed?	technological infrastructure will be required in the future?
Continuous implementation activities	How will we get from our current state to our desired	How are we going to implement the new	Transforming our software assets to the desired states?	How are we going to implement the required infra-structure?

49

Integration Perspectives

	Data	Process	Software	Infrastructure
Reverse Engineering Activities	Data Reverse Engineering	Business Process Reengineering	Software Reverse Engineering	Infrastructure Evaluation
Architecture Engineering Activities	Data Architecture Engineering		Software Architecture Development	Infrastructure Development/ Customization
Forward Engineering Activities	Data Evolution		Application Software Development	Infrastructure Modernization

50

Integration Perspectives

	Data	Process	Software	Infrastructure
Reverse Engineering Activities	Data Reverse Engineering	Business Process Reengineering	Software Reverse Engineering	Infrastructure Evaluation
Architecture Engineering Activities	Data Architecture Engineering		Software Architecture Development	Infrastructure Development/ Customization
Forward Engineering Activities	Data Evolution		Application Software Development	Infrastructure Modernization

51

Procedure Overview

Phase

1 What do you currently have?
   Reverse engineer existing assets by "as is" modeling

2 What would you like to have?
   Develop an architectural plan by modeling "to be" systems

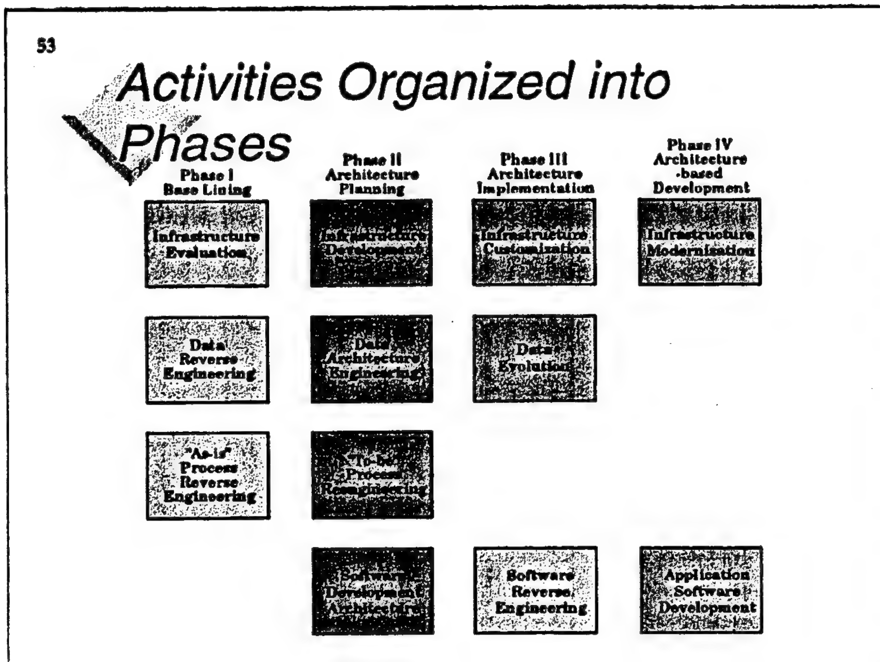
3 What do you need?
   Identify the gaps

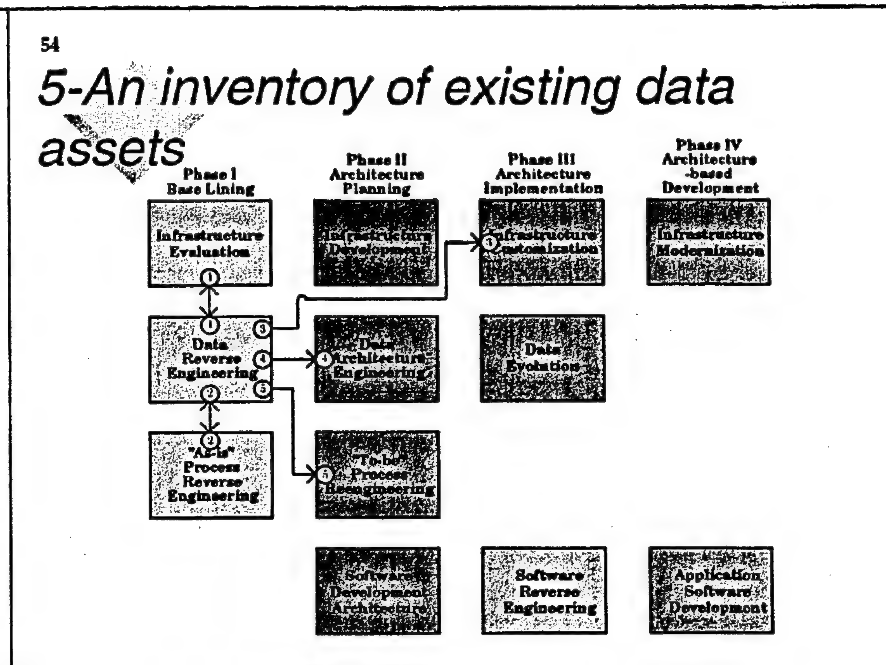
4 How do will you get there?
   Develop solutions to close the gaps and implement

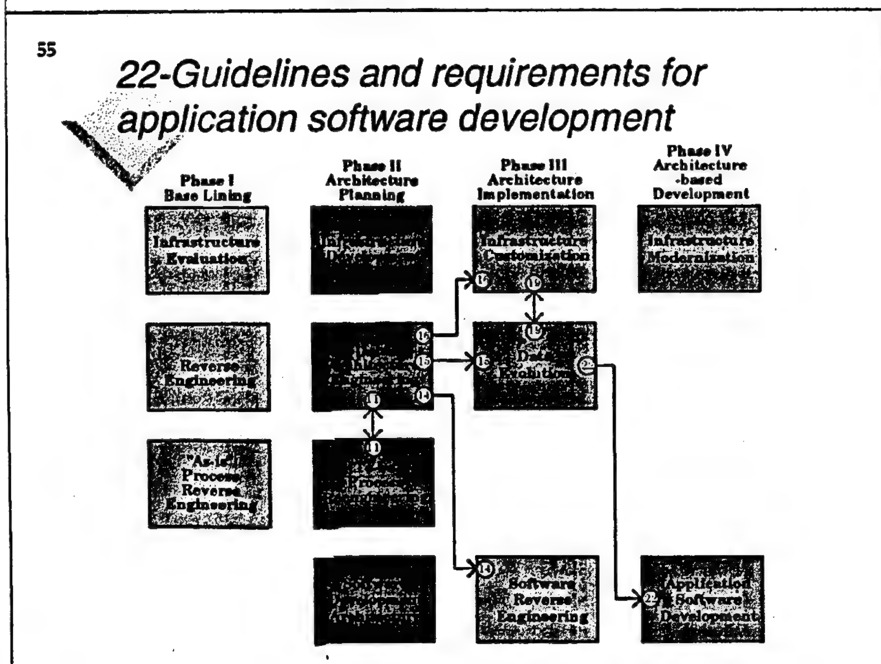
52

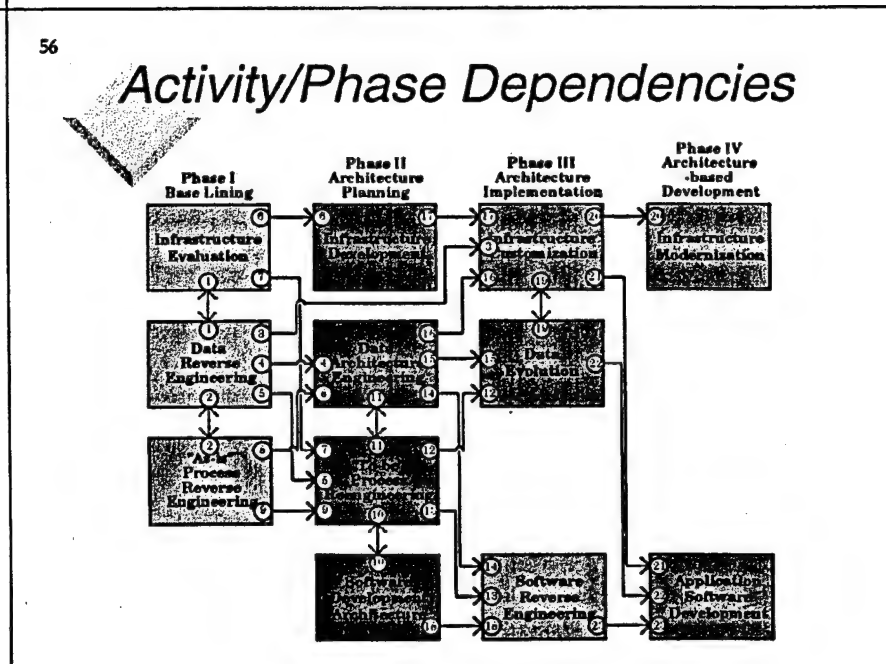
Phase Activity Relationships

Phase	Name	Phase Description	Associated Processes
1	Base Line Development Activities	The process of establishing an information base for further study and evaluation	• Data Reverse Engineering • Infrastructure Evaluation • "As-Is" Process Reverse Engineering
2	Architecture Planning Activities	Creation of plans guiding the subsequent development processes.	• "To-Be" Process Engineering • Data Architecture Engineering • Infrastructure Development • Software Architecture Development
3	Architecture Implementation Activities	Organization create working products based on the reengineered organizational architecture products.	• Infrastructure Customization • Data Evolution • Software Reverse Engineering
4	Architecture Population Activities	Create products capable of taking advantage of the features of the newly developed architecture.	• Infrastructure Modernization • Application Software Development


53
Activities Organized into Phases
Phase I Base Lining
Phase II Architecture Planning
Phase III Architecture Implementation
Phase IV Architecture -based Development
Infrastructure Evaluation
Infrastructure Customization
Infrastructure Modernization
Data Reverse Engineering
"As-is" Process Reverse Engineering
Software Reverse Engineering
Application Software Development


54
5-An inventory of existing data assets
Phase I Base Lining
Phase II Architecture Planning
Phase III Architecture Implementation
Phase IV Architecture -based Development
Infrastructure Evaluation
Infrastructure Customization
Infrastructure Modernization
Data Reverse Engineering
Data Architecture Engineering
Data Evolution
"As-is" Process Reverse Engineering
Software Development Architecture
Software Reverse Engineering
Application Software Development


55
22-Guidelines and requirements for application software development
Phase I Base Lining
Phase II Architecture Planning
Phase III Architecture Implementation
Phase IV Architecture -based Development
Infrastructure Evaluation
Infrastructure Customization
Infrastructure Modernization
Reverse Engineering
"As-is" Process Reverse Engineering
Software Reverse Engineering
Application Software Development


56
Activity/Phase Dependencies
Phase I Base Lining
Phase II Architecture Planning
Phase III Architecture Implementation
Phase IV Architecture -based Development
Infrastructure Evaluation
Infrastructure Customization
Infrastructure Modernization
Data Reverse Engineering
Data Evolution
"As-is" Process Reverse Engineering
Software Reverse Engineering
Application Software Development

57

Outputs

Data Reverse Engineering

1. Regular exchanges of information with any concurrent infrastructure evaluation processes
2. Data assets exchanged with "as-is" process reverse engineering efforts
3. System related technology constraints and opportunities
4. Validated data assets
5. An inventory of existing data assets

Infrastructure Evaluation

6. Infrastructure size, shape, growth rate and capacity
7. Infrastructure capabilities

"As-Is" Process Reverse Engineering

8. Data asset validation & description information
9. Base line business processes models

58

Outputs

"To-Be" Process Engineering

10. Software architectural guidance
11. Reengineered processes
12. Data inventories and required data transformations
13. Reengineered processes

Data Architecture Engineering

14. Data architecture-based data assets
15. Strategic guidance for data evolution
16. Data-based organizational infrastructure requirements

Infrastructure Development

17. Boundaries and/or standards

Software Architecture Development

18. Software architecture

59

Outputs

- *Infrastructure Customization*
- 19. Close coordination with any concurrent data evolution
- 20. Information required to guide subsequent infrastructure growth, evolution, and migration
- 21. Constraints and requirements for subsequent application software development processes
- *Data Evolution*
- 22. Guidelines and requirements for application software development
- *Software Reverse Engineering*
- 23. Reusable software assets

60

Milestone Descriptions

Phase I Milestone

validated business decompositions that is useful to a "to-be" process reengineering effort

Phase II Milestones

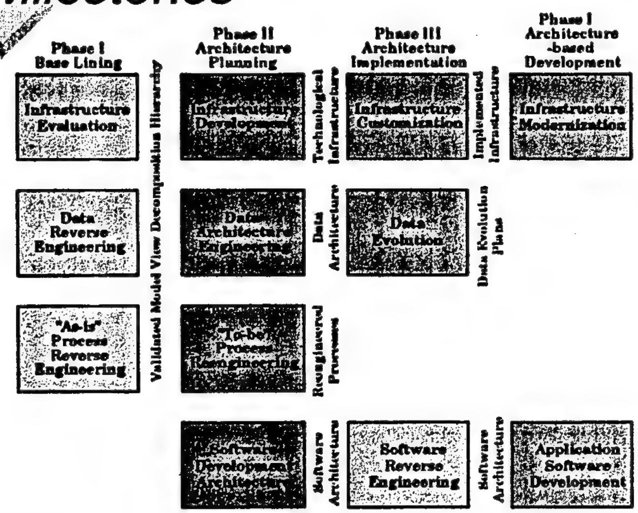
- Technological Infrastructure
- Data Architecture
- "To-be" Reengineered Processes
- Software Architecture

Phase III Milestones

- Suitable infrastructure for hosting the new application software developed
- Data evolution plans indicate specific destinations
- Reusing software assets from legacy systems in application software development processes
- Formats for evolving existing data to the new systems

61

Milestones



62

Q: Why Model?

- Models are used to help understand complex system behavior.
- Computer-based models are also an excellent means for storing and formalizing organizational information.
- Finally, models permit evaluation of various scenarios or other outcomes produced by the model.

63

Answers a set of specific questions

- What are the primary data objects to be processed by the system?
- What is the composition of each data object and what attributes describe the object?
- Where do objects currently reside?
- What are the relationships between each object and other objects?
- What is the relationship between the objects and the processes which transform them?

[PRES, 1992, p. 220]

64

Also ...

- Modeling is also used to filter out extraneous detail
- Model information can be considered basic or indispensable to understanding the system

65

Modeling Types

	Logical or Essential System	Physical or Implementation System
Current or Existing System	2 Logical "as-is"	1 Physical "as-is"
Proposed or Target System	3 Logical "to-be"	4 Physical "to-be"

66

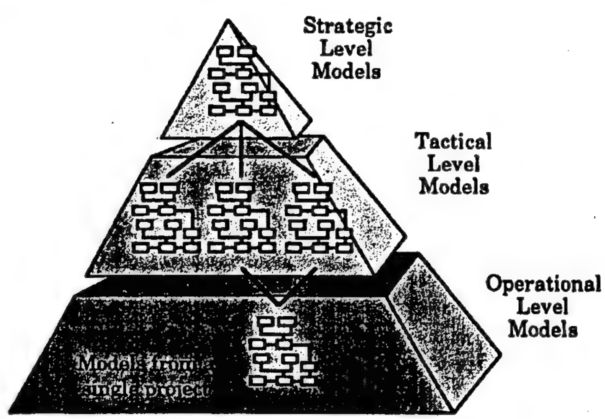
Why specify requirements in an implementation independent format?

Implementation independent models:

1. remove biases resulting from the current application or individuals - overcomes "we've always done it that way" syndrome - encourages creativity
2. reduce the risk of missing functional requirements because we are preoccupied with technical details - errors can be costly - separation of what from how permits better analysis for completeness, accuracy and consistency
3. allow us to communication with end-users in a non-technical language - avoids loosing communication through use of technical jargon

67

Information Modeling



This page intentionally left blank.

SOFTWARE REENGINEERING ASSESSMENT HANDBOOK (JLC-HDBK-SRAH)

Report to the
2nd SPC Reengineering Workshop

December 4-5, 1995

Robert E. Johnson, Jr.
Joint Logistics Commanders/
Computer Resources
Management (JLC/CRM),
SAM/AES Strategic C4 Plans
Pentagon
(703) 697-5397

John Clark
Comptek Federal Systems, Inc
Va Beach, Va.
(804) 463-8500

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

INTRODUCTION

- TO INTRODUCE JLC-HDBK-SRAH, VERSION 2.0, MARCH 1995
  - A QUICK METHOD TO DETERMINE IF REENGINEERING IS NEEDED AND COST EFFECTIVE
- THREE SEQUENTIAL PROCESSES:
  - TECHNICAL ASSESSMENT: EVALUATE SOFTWARE CANDIDATES AND SELECT REENGINEERING STRATEGIES
  - ECONOMIC ASSESSMENT: CALCULATE ECONOMIC INDICATORS FOR EACH STRATEGY OF EACH CANDIDATE
  - MANAGEMENT DECISION: EVALUATE, SELECT, AND PRIORITIZE CANDIDATES AND THEIR STRATEGIES

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

TECHNICAL REPORT

SOFTWARE REENGINEERING ASSESSMENT HANDBOOK

Version 2.0

Volumes I and II

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

JLC-HDBK-SRAH BACKGROUND

- JLC-JPCG-CRM WORKING GROUP AT SB-1, TRI-SERVICE
- VERSION 1.0 BY COMPTEK/MCR
  - UNDER COGNIZANCE OF:
    - AIR FORCE COST ANALYSIS AGENCY
    - SOFTWARE TECHNOLOGY SUPPORT CENTER, HILL AFB
  - TECHNICAL PARTICIPATION BY:
    - AIR FORCE STANDARD SYSTEMS CENTER, GUNTER AFB
    - COST MODEL DEVELOPERS
  - FOUR FIELD TESTS CONDUCTED
  - RELEASED FOR BROAD COMMUNITY REVIEW FEB 94
- VERSION 2.0 BY COMPTEK/SAIC/STSC
  - 70 SETS OF COMMENTS INCORPORATED
  - RELEASED APRIL 95 AT STC ON THE CD ROM
  - NOW A JLC-JPCG-CRM PRODUCT

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

APPLICABILITY

- DOMAIN:
  - VARIETY OF SOFTWARE (AIS AND TACTICAL/REAL-TIME)
  - VARIOUS LEVELS OF AN ORGANIZATION
  - DoD, NON-DoD, COMMERCIAL, INDUSTRIAL, ACADEMIC
  - MAINTENANCE, REUSE, COTS/NDI IN NEW/EXISTING SYSTEMS
- CASES:
  - 1: A SPECIFIC CANDIDATE WITH A SINGLE STRATEGY
  - 2: A SPECIFIC CANDIDATE WITH MULTIPLE STRATEGIES
  - 3: A SET OF CANDIDATES EACH WITH MULTIPLE STRATEGIES
- CHOICES:
  - MAINTAIN STATUS QUO
  - REENGINEER
  - RETIRE

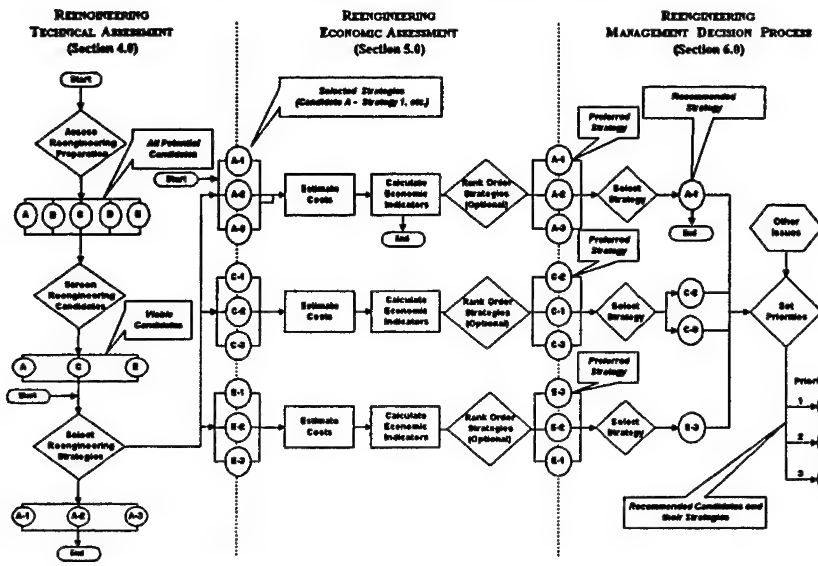
2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

STRATEGIES

- STATUS QUO (ALWAYS STRATEGY 1)
- REVERSE ENGINEERING
- RESTRUCTURING
- TRANSLATION
- DATA REENGINEERING
- REDOCUMENTATION
- FORWARD ENGINEERING
- RETARGETING
- REDEVELOPMENT
- ARCHITECTURE TRANSFORMATION (UNDER CONSIDERATION)

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

OVERALL PROCESS



Note: A-1 indicates a candidate-strategy pair wherein 1 indicates Status Quo

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

TECHNICAL ASSESSMENT PROCESS (STRATEGY SELECTION)

- Purpose: To match legacy software components with reengineering strategies.
- Gives a rough idea of where reengineering can help maintenance activities for those organizations unaware of reengineering principles.
- Based on:
  - JLS's Santa Barbara - I Reengineering Workshop (Sept. 1992)
  - USAF organization interviews
  - Fiels tests of SRAH version 1.0
    - Gunter AFB, AL
    - Wright-Patterson AFB, OH
    - Lawrence Livermore Labs, CA
    - Hill AFB, UT
  - Initial reengineering survey results
  - Other reengineering projects data

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

STRATEGY SELECTION (cont.)

- Reengineering Projects Data Repository
  - SRAH validation, modification, and enhancement
  - Will help with version 3.0 issues of
    - "Weighting" of questions
    - Software size issues
  - Pre and Post Surveys with Instruction Set
- 6 Reengineering Strategies
  - Redocument
  - Reverse Engineer
  - Translate Source Code
  - Data Reengineer
  - Restructure
  - Retarget
- 2 Classical Maintenance Strategies
  - Redevelopment
  - Status Quo

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

STRATEGY SELECTION (cont.)

- Step 1: Assess Preparedness (Question Set)
- Step 2: Identify Software Candidates
  - Consider factors of age, complexity, language, reliability, HW/SW coupling, platform changes, etc.
- Step 3: Reduce List of Software Candidates
  - Suggest remove from list if:
    - Remaining life < 3 years
    - Not important enough (Importance question set)
    - Age < 5 years
    - Software directly supports ongoing BPR efforts
- Step 4: Complete Strategy Selection Question Sets
  - Redocument
  - Restructure
  - Translate Source Code
  - Data Reengineer
  - Retarget

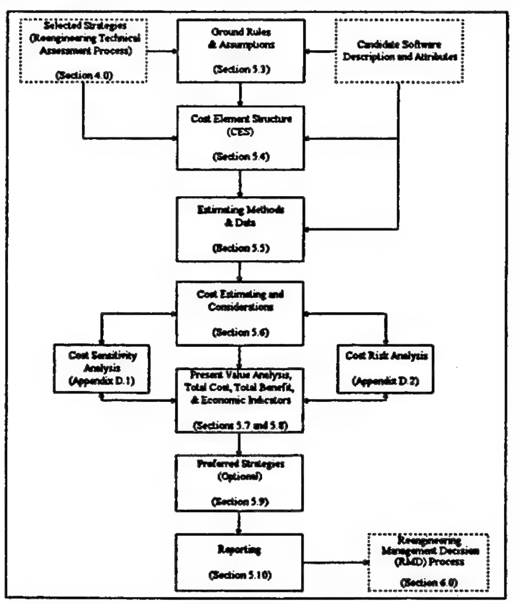
2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

STRATEGY SELECTION (cont.)

- Step 5: Consider Other Maintenance Strategies
  - Reverse Engineer if:
    - Multiple reengineering strategies are indicated
    - Highly complex control flow
    - Reuse is a key objective
  - Redevelopment if:
    - 3 or more reengineering strategies indicated
    - Remaining life > 5 years
  - Status Quo if:
    - No reengineering strategies are indivcated
- Maintenance Environment Considerations (Question Set)
- Misc. Other Issues
  - Impotrance of Pilot Project
  - Impact Analysis
  - Pitfalls of Source Code Translation and Restructuring
  - Software Analysis Tools

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

ECONOMIC ASSESSMENT PROCESS



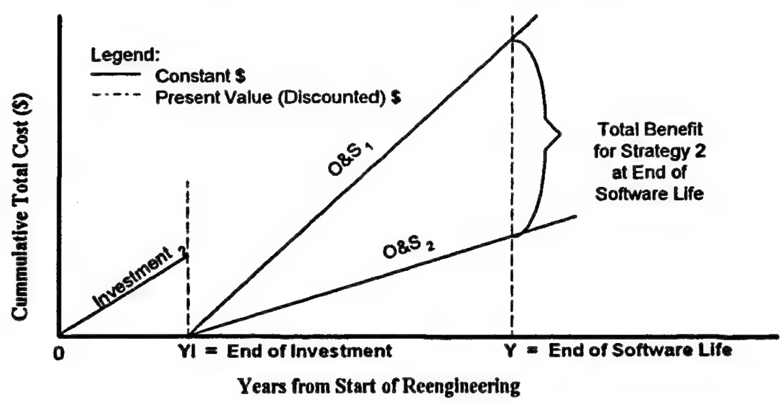
2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

ECONOMIC TERMINOLOGY

- BENEFIT (QUANTIFIABLE AND NON-QUANTIFIABLE)
- BENEFIT INVESTMENT RATIO (BIR): 2ND BEST INDICATOR
- BREAK-EVEN POINT (BP)
- COST SAVINGS AND COST AVOIDANCE
- CONSTANT DOLLARS
- CURRENT DOLLARS
- DISCOUNTED DOLLARS
- INVESTMENT COST AND O&S COST
- NET VALUE (NV)
- PRESENT VALUE (PV)
- NET PRESENT VALUE (NPV): BEST INDICATOR
- RATE OF RETURN (ROR)
- SUNK COST
- UNIFORM ANNUAL COST

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

NET VALUE AND BIR



NET VALUE = BENEFIT - INVESTMENT

= $O\&S_1$ - $O\&S_N$ - INVESTMENT

> 0 : POSITIVE RETURN

< 0 : NEGATIVE RETURN

= 0 : BREAK-EVEN

BIR = BENEFIT / INVESTMENT

= ($O\&S_1$ - $O\&S_N$) / INVESTMENT

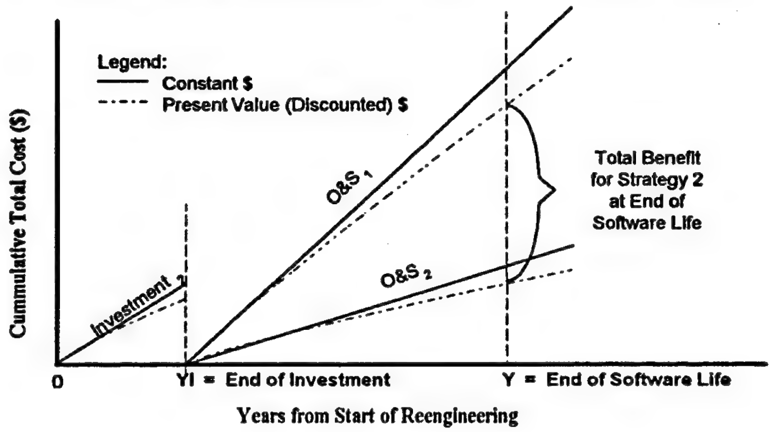
> 1 : POSITIVE RETURN

< 1 : NEGATIVE RETURN

= 1 : BREAK-EVEN

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

NET PRESENT VALUE AND BIR



NET PRESENT VALUE = PV (BENEFIT) - PV (INVESTMENT)

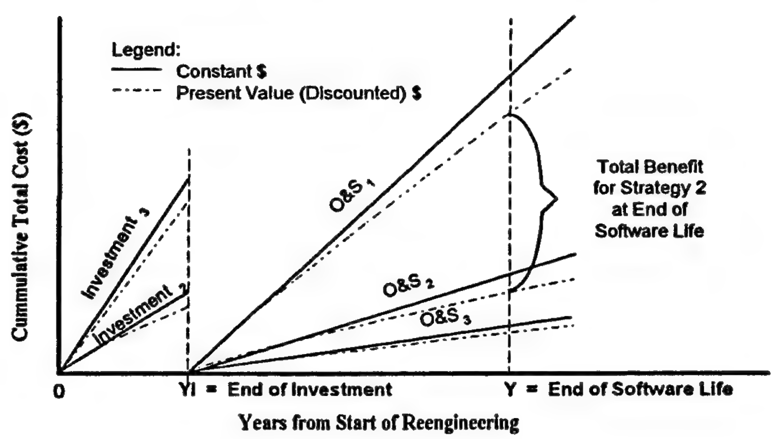
= PV ($O\&S_1$) - PV ($O\&S_N$) - PV (INVESTMENT)

BENEFIT INVESTMENT RATIO = PV (BENEFIT) / PV (INVESTMENT)

= (PV ($O\&S_1$) - PV ($O\&S_N$)) / PV (INVESTMENT)

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

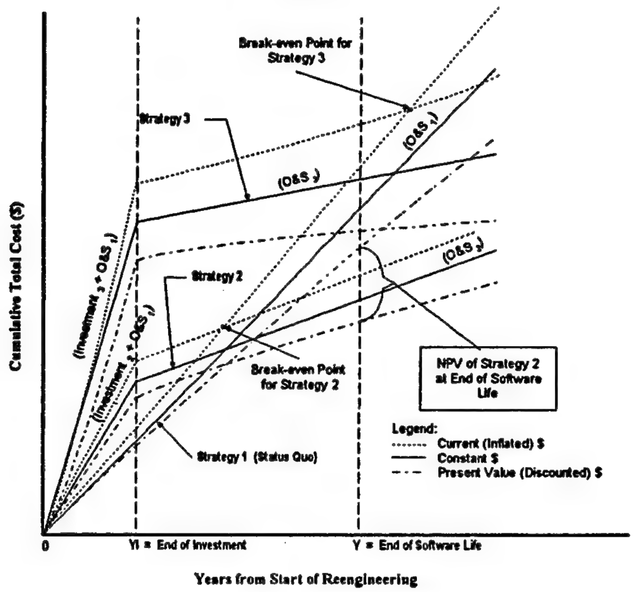
STRATEGY SELECTION



WHICH STRATEGY IS BEST: 1, 2 OR 3 ?

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

TOTAL COST AND BP

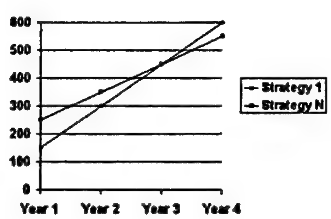


This page intentionally left blank.

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

SIMPLIFIED EXAMPLE
(no discounting or inflation)

Cost Type	Year	Cost of Status Quo	Cost of Strategy N	Benefit
Investment	1	NA	100	NA
O&S	1	150	150	0
O&S	2	150	100	50
O&S	3	150	100	50
O&S	4	150	100	50
Total		600	550	150



NV = 150 - 100 = 50 or 600 - 550 = 50, BIR = 150 / 100 = 1.5

BP = Year 3 when cumulative benefit = investment = 100

ROR = i = 0.234 or 23.4 % when:

$$\frac{100}{(1+i)^1} = \frac{0}{(1+i)^1} + \frac{50}{(1+i)^2} + \frac{50}{(1+i)^3} + \frac{50}{(1+i)^4}$$

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

REENG INVESTMENT WORKSHEET

PROGRAM ______________ REENGINEERING INVESTMENT COST (CES 1.0) WORKSHEET PAGE ___ OF ___
STRATEGY: ______________ CONSTANT FY _____ $K, DISCOUNT RATE _____ %, INFLATION RATE _____ % ANALYST ____ DATE ______

CES COST ELEMENT	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	TOTAL	DATA SOURCE/NOTES
1.1 Software Development												
1.1.1 Requmts. Analysis												
1.1.2 Prelim. Design												
1.1.3 Detailed Design												
1.1.4 Code & Unit Test												
1.1.5 Unit Integ. & Test												
1.1.6 CSCI Test												
1.1.7 SPCR Resolution												
1.1.8 Reverse Engr.												
1.1.9 Other												
1.2 CSCI-CSCI Integ. & Test												
1.3 System Integ. & Test												
1.4 Training												
1.5 Data												
1.6 Peculiar Supt. Equip.												
1.7 Operational Site Activ.												
1.8 Facilities & Utilities												
1.9 Hardware												
1.10 System Operations												
1.11 IV & V												
1.12 Sys. Eng./Pgm. Mgmt.												
1.13 Other												
1.0 Investment (Constant $)												
Mid-Yr Discount Factor												= Cell D1
1.0 Investment (Present Value)												= Cell T1
1.0 Investment (Current $)												

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

REENG O&S WORKSHEET

PROGRAM ______________ REENGINEERING O&S COST (CES 2.0) WORKSHEET PAGE ___ OF ___
STRATEGY: ______________ CONSTANT FY _____ $K, DISCOUNT RATE _____ % INFLATION RATE _____ % ANALYST ____ DATE ______

CES COST ELEMENT	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	TOTAL	DATA SOURCE/NOTES
2.1 Software Support												
2.1.1 Requmts. Analysis												
2.1.2 Prelim. Design												
2.1.3 Detailed Design												
2.1.4 Code & Unit Test												
2.1.5 Unit Integ. & Test												
2.1.6 CSCI Test												
2.1.7 SPCR Resolution												
2.1.8 Other												
2.2 CSCI-CSCI Integ. & Test												
2.3 System Integ. & Test												
2.4 Training												
2.5 Data												
2.6 Peculiar Supt. Equip.												
2.7 Operational Site Activ.												
2.8 Facilities & Utilities												
2.9 Hardware												
2.10 System Operations												
2.11 IV & V												
2.12 Sys. Eng./Pgm. Mgmt.												
2.13 Other												
2.14 Existing O&S												Table B-3, CES 3.0 for Y1 only
2.0 O&S (Constant $)												
Mid-Yr Discount Factor												= Cell D2
2.0 O&S (Present Value)												= Cell T2
2.0 O&S (Current $)												

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

STATUS QUO WORKSHEET

PROGRAM ______________ EXISTING O&S COST (CES 3.0) WORKSHEET PAGE___ OF___

STRATEGY: ______________ CONSTANT FY _____ $K, DISCOUNT RATE_____%, INFLATION RATE_____% ANALYST____ DATE______

CES COST ELEMENT	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	TOTAL	DATA SOURCE/NOTES
3.1 Software Support												
3.1.1 Requmts. Analysis												
3.1.2 Prelim. Design												
3.1.3 Detailed Design												
3.1.4 Code & Unit Test												
3.1.5 Unit Integ. & Test												
3.1.6 CSCI Test												
3.1.7 SPCR Resolution												
3.1.8 Other												
3.2 CSCI-CSCI Integ. & Test												
3.3 System Integ. & Test												
3.4 Training												
3.5 Data												
3.6 Peculiar Supt. Equip.												
3.7 Operational Site Activ.												
3.8 Facilities & Utilities												
3.9 Hardware												
3.10 System Operations												
3.11 IV & V												
3.12 Sys. Eng./Pgm. Mgmt.												
3.13 Other												
3.0 O&S (Constant $)												
Mid-Yr Discount Factor												= Cell D8
3.0 O&S (Present Value)												= Cell T3
3.0 O&S (Current $)												

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

BREAK-EVEN WORKSHEET

PROGRAM ______________ BREAK-EVEN ANALYSIS WORKSHEET (CURRENT $) PAGE___ OF___

STRATEGY: ______________ CONSTANT FY _____ $K, INFLATION RATE_____% ANALYST____ DATE______

CES COST ELEMENT	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	FY__	TOTAL	DATA SOURCE/NOTES
A. ANNUAL COSTS												
3.0 Existing O&S (CES 3.0)												Table B-3, O&S (Current $)
2.0 Reengineering O&S (CES 2.0)												Table B-2, O&S (Current $)
1.0 Reengineering Invest. (CES 1.0)												Table B-1, Investment (Current $)
Total Reeng. Cost (1.0 + 2.0)												Sum of 1.0 and 2.0 Above.
B. CUMULATIVE COSTS												
3.0 Existing O&S (CES 3.0)												Cumulative of CES 3.0
2.0 Reengineering O&S (CES 2.0)												Cumulative of CES 2.0
1.0 Reengineering Invest. (CES 1.0)												Cumulative of CES 1.0
Total Reeng. Cost (1.0 + 2.0)												Sum of 1.0 and 2.0 Above.
C. BREAK-EVEN POINT												
Existing O&S - Total Reeng. Cost												Cumulative of 3.0 - 2.0 - 1.0
Break-even Year												BP when Existing - Reeng. = 0

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

SUMMARY WORKSHEET

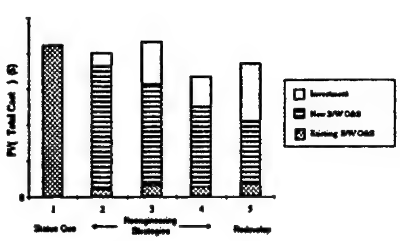
PROGRAM: ______________ **ECONOMIC ASSESSMENT SUMMARY WORKSHEET** **PAGE:** ____ **OF** ____

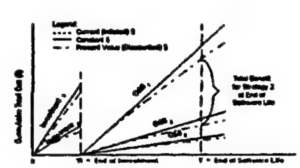
ANALYST: ______________ **TOTAL COST** ____ **OR UNIFORM ANNUAL COST** ____ **DATE:** ______

METRICS	REENGINEERING STRATEGIES								
	2	3	4	5	6	7	8	9	DATA SOURCE/NOTES
A. PARAMETERS									
A1 SLOC/FP									Counted or Estimated
A2 ESLOC/EFP									Counted or Estimated
A3 Annual Change Traffic (ACT %)									Calculated or Estimated
A4 Remaining Life Years (Y)									Specified
A5 Investment Years (YI)									Estimated
A6 Support Years (YS)									YS = Y - YI
B. DISCOUNTED FY__ $									
B1 PV Total O&S (Strategy 1)									Table B-3, Cell T3
B2 PV Total O&S (Strategy N)									Table B-2, Cell T2 for Each Strategy
B3 PV Total Benefit (TB)									B1 - B2 for Each Strategy
Rank Order									
B4 PV Total Investment (TI)									Table B-1, Cell T1 for Each Strategy
Rank Order									
C. ECONOMIC INDICATORS									
C1 Net Present Value (NPV)									B3 - B4 for Each Alternative
Rank Order									
C2 Benefit Investment Ratio (BIR)									B3 / B4 for Each Alternative
Rank Order									
C3 PV Total Cost (TC)									B2 + B4 for Each Strategy
Rank Order									
C4 Break-even Point (Yrs)									Year when Current TC(1) = TC(N)
Rank Order									
C5 Rate-of-Return (%)									Interest Rate when NPV = 0
Rank Order									

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

ECONOMIC REPORT





Section	Description
1.0	Overview
1.1	Background
1.2	Scope
1.3	Major Assumptions
1.4	Major Constraints
2.0	Alternatives
3.0	Summary
3.1	Cost Summary
3.2	Benefit Summary
3.3	Sensitivity Analysis Summary
3.4	Risk Analysis Summary
3.5	Summary of Recommendations
3.6	Program/Project Mgmt Charter
4.0	Analyses
4.1	Benefit Analysis
4.2	Sensitivity Analysis
4.3	Risk Analysis
4.4	Conclusion
4.5	Cost Data Sheets
4.6	Variable Explanation Sheets

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

COST MODELS (VOLUME II)

- RESIZE COMPTEK: JOHN CLARK, MIKE WOOD
- REVIC (TO BE SUPPLIED)
- PRICE S MARTIN MARIETTA PRICE S: JIM OTTE
- SEER-SEM GALORATH: ALAN CLARK, KAREN MC RITCHIE
- SLIM QSM: DOUG PUTNAM, LARRY PUTNAM JR.
- SOFTCOST-OO RESOURCE CALCULATIONS: TONY COLLINS
- CHECKPOINT SW PROD. RESEARCH: CAPERS JONES

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

MANAGEMENT DECISION PROCESS

- Purpose: To combine quantitative data fron the Technical Assessmsnt Process, Economic Assessment Process, and subjective organizational data into a repertable format for consistent reengineering project implementation decisions
- Quantitative vs. Subjective Decision Criteria
  - No direct correlation established between quantitative data and optimum strategy
  - Create a balance between these two issues
- Decision Process consists of:
  - Management Report Preparation
  - Decision Process
  - Decision Documentation

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

MANAGEMENT DECISION (cont.)

- Management Report Preparation
  - Complete Detailed Assessment Results Worksheet
  - System Information
  - Strategy Information
  - Recommended Ranking
- Executive Overview
  - Recommended Strategy Rank Worksheet
  - Ranking Explanation
  - Report Introduction
- Decision Process
  - Additional organizational factors
  - Making the decision
- Decision Documentation
  - Archive results for organizational calibration of SRAH

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

REENGINEERING ASSESSMENT SERVICES

- John Clark
  - Comptek Federal Systems, 2877 Guardian Lane, VA Beach VA 23452
    (804) 463-8500 clark@comptek.com
- Mike Olsem
  - STSC / SAIC, OOALC/TISEC, 7278 4th St., Hill AFB, UT 84056-5205
    (801) 777-5555 ext 3057 or (801) 825-2655 olsemm@software.hill.af.mil

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

CURRENT ACTIVITIES

- NRaD, San Diego, CA
- NSWCDD, White Oak, MD
- NUWCDIV, Newport, RI
- PMO450 & PMO401, Washington DC

2nd SPC REENGINEERING WORKSHOP *JLC-HDBK-SRAH*

SUMMARY

- JLC-HDBK-SRAH VERSION 2.0 RELEASED FOR USE
- PLACED ON THE CD AT THE STC IN APRIL '95
- TO RECEIVE A HARDCOPY:
  - SOFTWARE TECHNOLOGY SUPPORT CENTER
    - CUSTOMER SUPPORT: (801) 777-8045
- SEND COMMENTS TO:
  - ROBERT E. JOHNSON, Jr.
    SAM/AES Strategic C4 Planning
    PENTAGON ROOM 1D148
    WASHINGTON, DC 20310-0107

    VOICE: (703) 697-5397
    FAX: (703) 697-3477
    EMAIL: johnsonr@comm.hq.af.mil

This page intentionally left blank.

Software Productivity Consortium
2nd SPC Reengineering Workshop
December 4, 1995

CLASSIFYING TOOLS FOR REENGINEERING: A STATE OF THE INDUSTRY

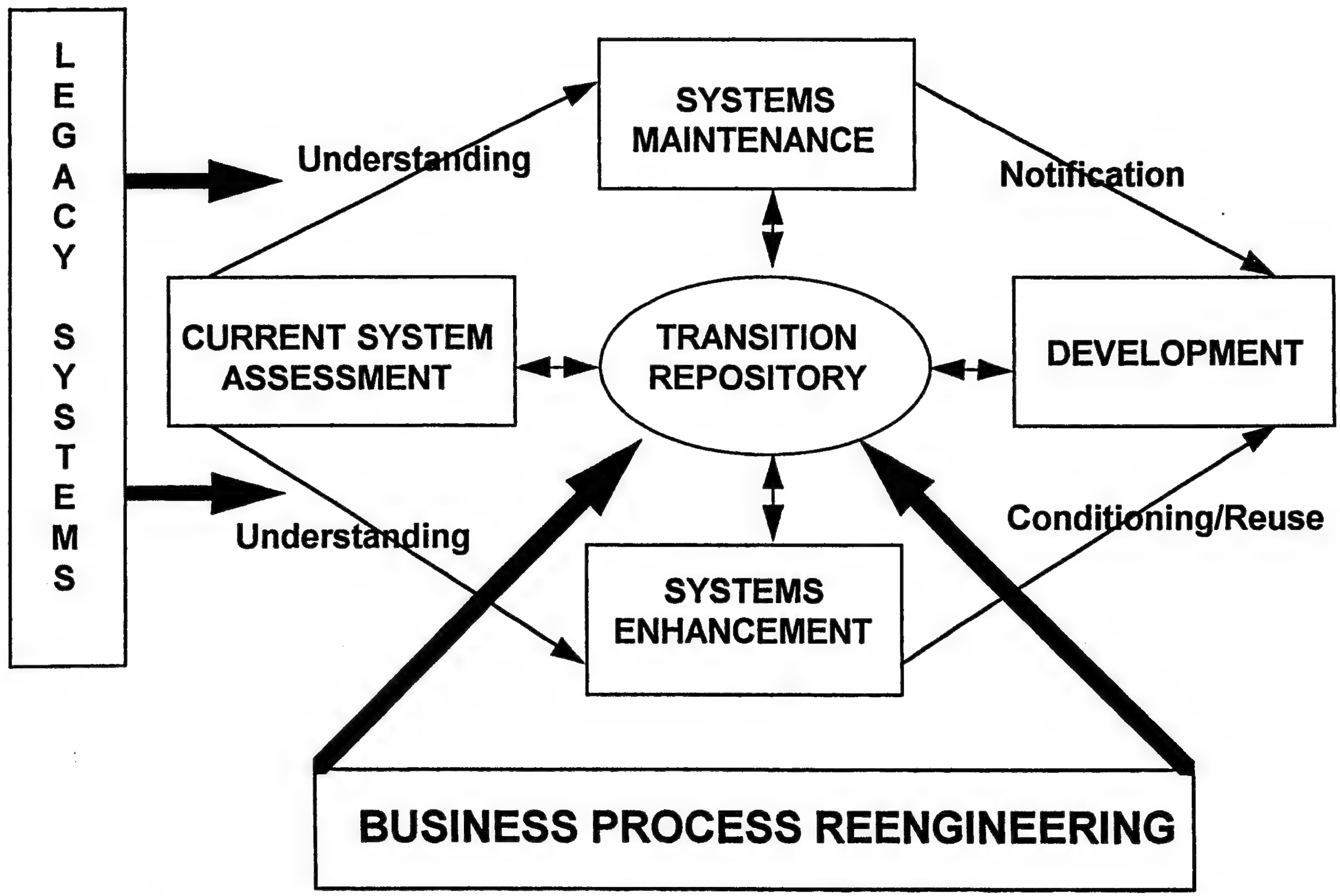
Presented By:
David Sharon
CASE Associates Inc.
14915 SE 82nd Drive
Clackamas, OR 97015
(503) 656-0986

WM. DAVID SHARON
PRESIDENT
CASE ASSOCIATES INC.

A Member of the Spectrum Institute

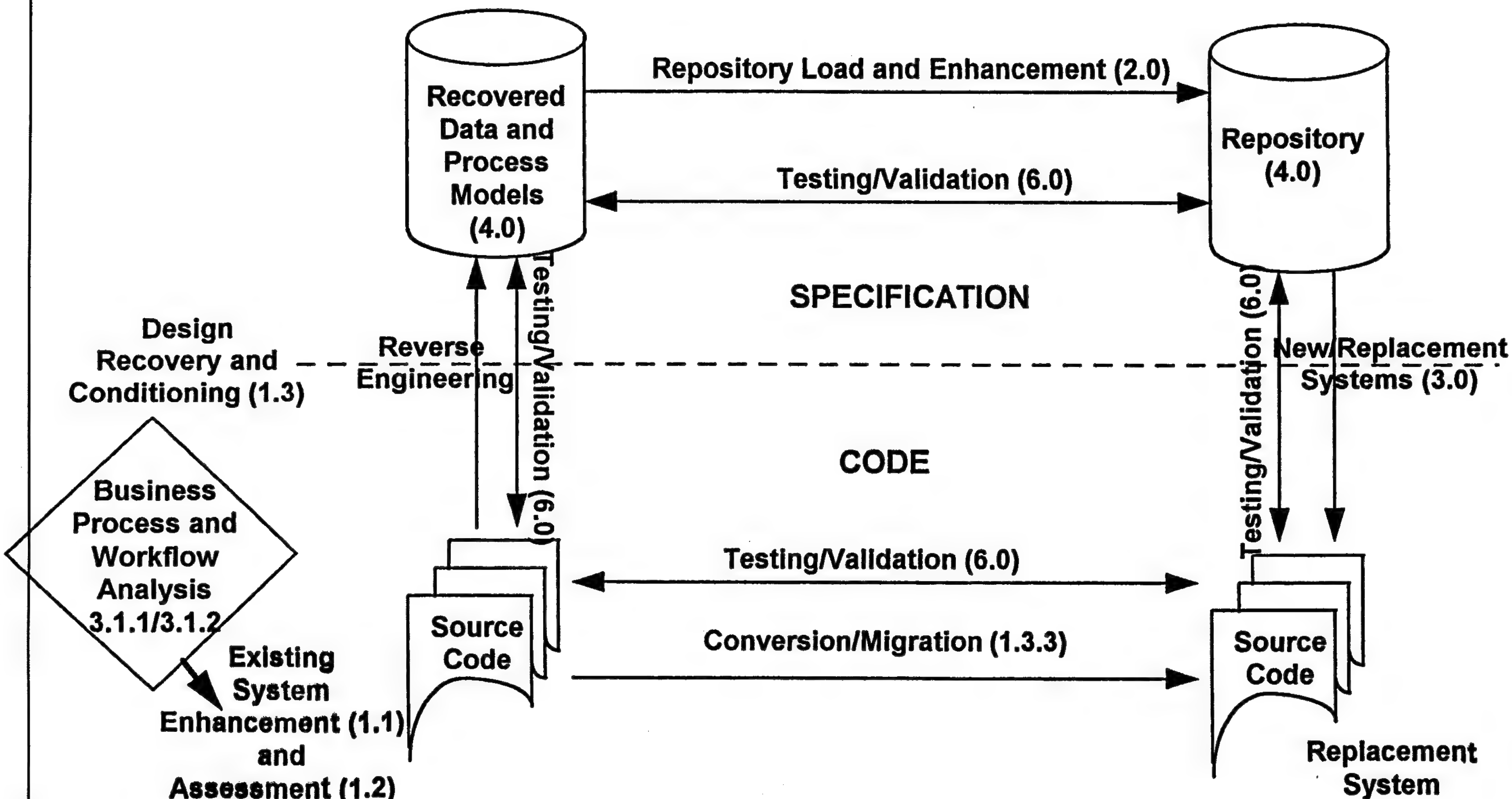
- Over 28 years of software industry experience
- The CASE Industry Advisor to Software Magazine
- Researcher and Publisher of the CASE Locator®, CASE Buyer's Guide®, and CASE Almanac®
- Contributing Editor to Application Development Trends
- Tool Box Column Editor for IEEE Software
- Author of a soon-to-be-published book Managing Systems in Transition and of CAI's Business/System Process Improvement Program Developer®
- Since 1991, a Judge for the ITAA Quality Award Program
- On Board of Directors of the Systems Development Forum
- Current Secretary of the IEEE Task Force on Professional Tools
- A Founder and Former Editor/Market Analyst of the CASE Outlook
- Former Director of Product Marketing, Nastec Corporation and Former Marketing Manager, CASE Division, Tektronix, Inc.
- BS from University of California, Berkeley, and MBA from Portland State University

THE RELATIONSHIP OF SYSTEMS MAINTENANCE, ENHANCEMENT, AND DEVELOPMENT



 9

THE RELATIONSHIP BETWEEN APPLICATION DEVELOPMENT, MAINTENANCE, AND REENGINEERING



10

THE MAJOR CLASSES

1.0 EXISTING SYSTEMS
2.0 REPOSITORY LOAD/ENHANCEMENT
3.0 NEW/REPLACEMENT SYSTEMS
4.0 REPOSITORIES
5.0 INTEGRATED TOOL SET ENVIRONMENTS
6.0 TESTING/VALIDATION
7.0 SOFTWARE/PROJECT MANAGEMENT
8.0 DBMS/NETWORK/FILE MANAGEMENT
0.0 MISCELLANEOUS

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

- 1.0 Existing systems
  - 1.1 Enhancement
    - 1.1.1 Smart editors/browsers
    - 1.1.2 Maintenance environment
  - 1.2 Assessment
    - 1.2.1 Measurement
    - 1.2.2 Inventory/analysis
    - 1.2.3 Redocumentation
  - 1.3 Conditioning
    - 1.3.1 Data rationalization
    - 1.3.2 Process rationalization
    - 1.3.3 Conversion/migration
    - 1.3.4 Code (process) restructuring

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

2.0 Repository load/enhancement and reconciliation
- 2.1 Data
- 2.2 Process

3.0 New/replacement systems
- 3.1 Planning
  - 3.1.1 Modeling/BPA/workflow analysis
  - 3.1.2 Strategic planning
- 3.2 Analysis/design
  - 3.2.1 SA/SD
  - 3.2.2 OOA/OOD
  - 3.2.3 Other models

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

3.3 Construction
- 3.3.1 Code generators (non-OO)
- 3.3.2 OO language tools
- 3.3.3 Visual programming tools
- 3.3.4 4GLs
- 3.3.5 Compilers

3.4 GUI builder

3.5 Prototyping/simulation

3.9 MetaCASE

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

- 4.0 Repositories
  - 4.1 Repositories/data dictionaries
  - 4.2 Repository/data-dictionary manager
    - 4.2.1 Object management systems
    - 4.2.2 Reuse management systems
  - 4.3 Data warehouse

23

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

5.0 Integrated toolset environments
- 5.1 Integration frameworks
- 5.2 Integration utilities
- 5.3 Resulting integrated tools
- 5.4 ICASE tools

6.0 Testing validation
- 6.1 Test-planning and management
- 6.2 Test-data generation
- 6.3 Execution/testing
- 6.4 Capture/playback
- 6.5 Coverage analysis
- 6.6 Validation/correction
- 6.7 Code/data comparison
- 6.8 GUI testers

24

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

7.0 Software/project management
- 7.1 Process workbenches/managers
- 7.2 Process management methodology
- 7.3 Project management
- 7.4 Estimation/projection
- 7.5 Job accounting/chargeback
- 7.6 Performance management
- 7.7 Problem tracking
- 7.8 Configuration management
- 7.9 Document management
- 7.10 Requirements management
- 7.11 Operations management
- 7.12 Training
- 7.13 Acquisition/contract management

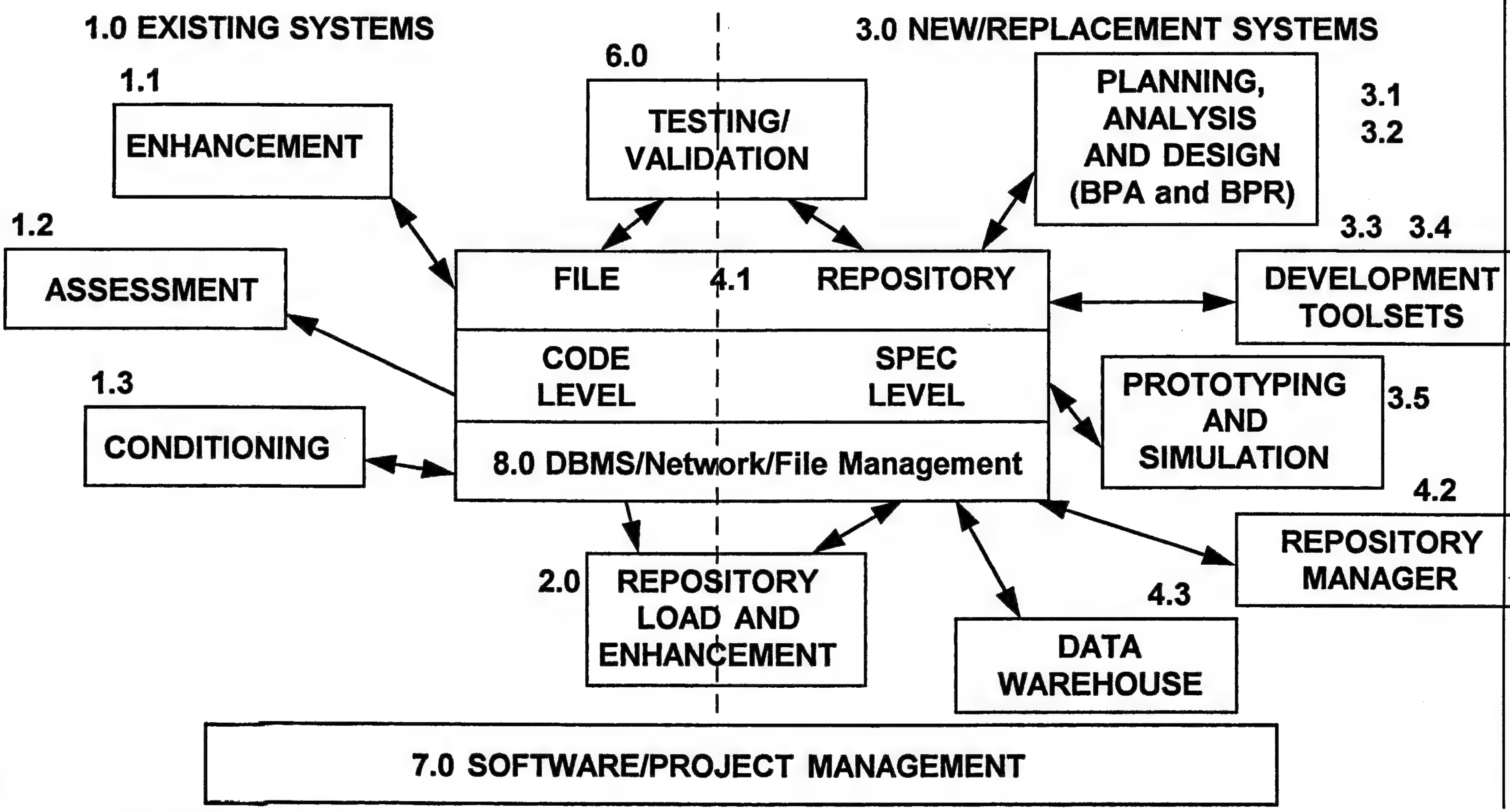
APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION SCHEME

8.0 DBMS/network/file management
- 8.1 Database management systems
- 8.2 Network/communication/file managers
- 8.3 Middleware

0.0 Miscellaneous

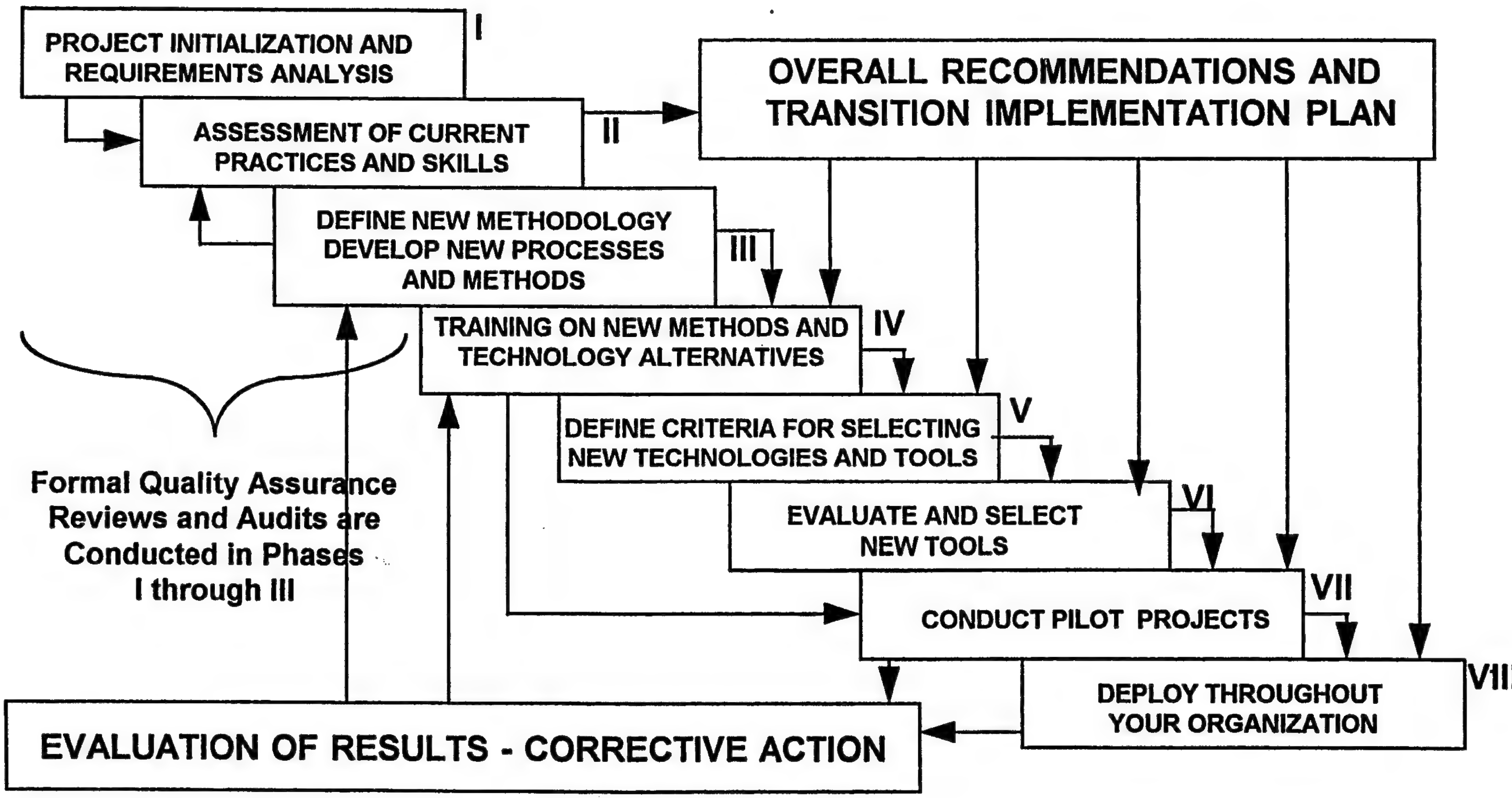
26

APPLICATION DEVELOPMENT, REENGINEERING, AND MAINTENANCE TOOL CLASSIFICATION RELATIONSHIPS



27

BUSINESS/SYSTEM PROCESS IMPROVEMENT PROGRAM



37

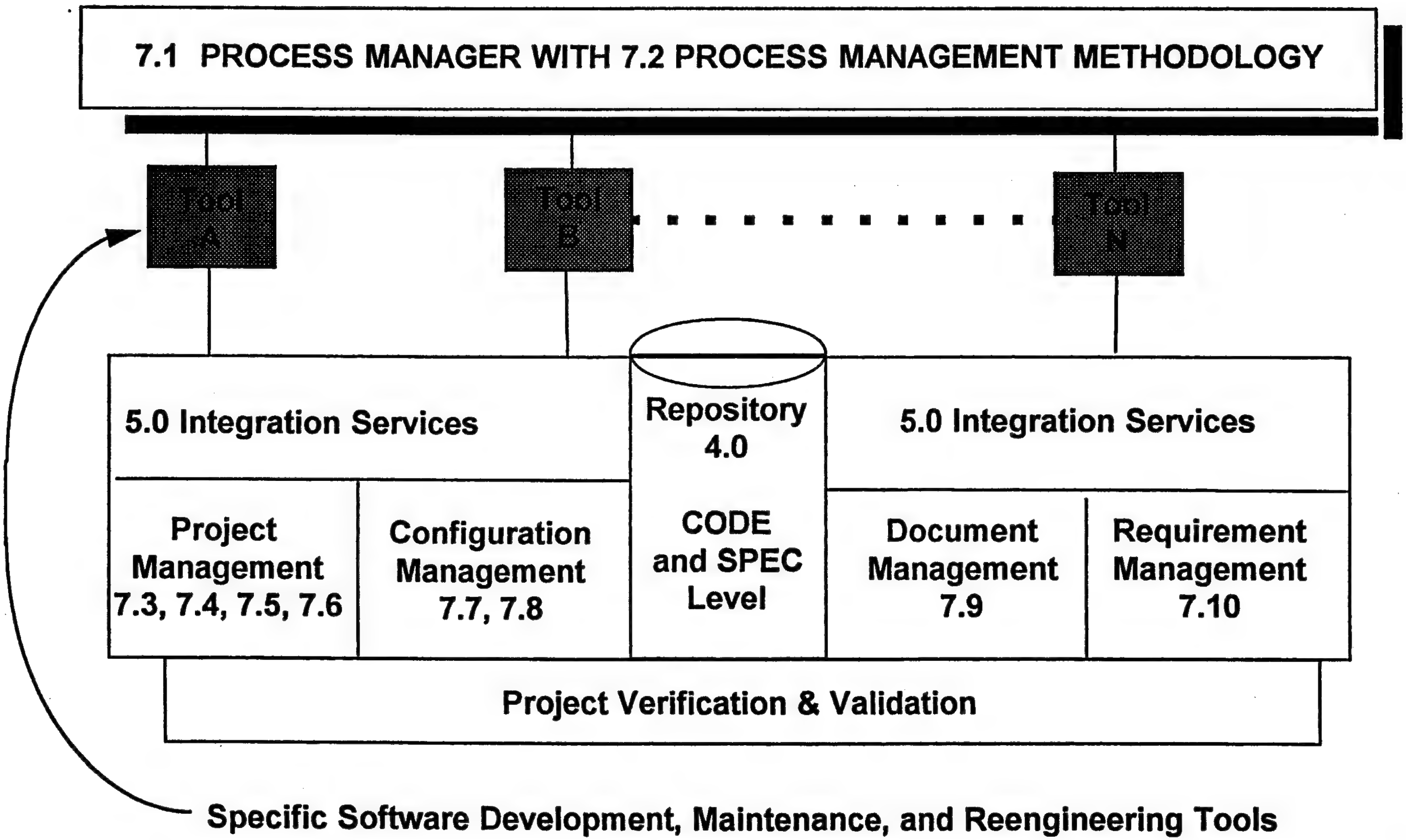
TOOLS AND SOLUTIONS FOR MANAGING SYSTEMS TRANSITIONS

Implement the fundamentals of engineering first around a repository

- Configuration management
- Project management
- Process management
- Team coordination and information sharing
- Project verification and validation
- Documentation management

40

A FRAMEWORK FOR SOFTWARE ENGINEERING MANAGEMENT



41

Appendix A

TOOL CLASSIFICATION SCHEME DEFINITIONS

1.0 EXISTING SYSTEMS

- 1.1 ENHANCEMENT
  - 1.1.1 SMART EDITORS/BROWSERS
  - 1.1.2 MAINTENANCE ENVIRONMENT
- 1.2 ASSESSMENT
  - 1.2.1 MEASUREMENT
  - 1.2.2. INVENTORY/ANALYSIS
  - 1.2.3 RE-DOCUMENTATION
- 1.3 CONDITIONING
  - 1.3.1 DATA RATIONALIZATION
  - 1.3.2 PROCESS RATIONALIZATION
  - 1.3.3 CONVERSION/MIGRATION
  - 1.3.4 CODE (PROCESS) RESTRUCTURING

Examine existing system components at the code level to provide information about the software for making changes (maintenance) and current system confirmation for new development (software verification and validation)

1.1 ENHANCEMENT - for understanding an existing system before making changes

1.1.1 SMART EDITORS/BROWSERS - for analysis of structure, organization, data usage and relationships and logical execution paths within one or more programs.

1.1.2 MAINTENANCE ENVIRONMENT - combine the capabilities of smart editors and browsers with a code level repository for analysis in or between programs, screens, JCL, databases, and TP monitors.

1.2 **ASSESSMENT** - examine the source code components of existing systems by measuring the characteristics of the components, creating an inventory and classification of the components, and creating documentation. This is a discovery process during which no changes are made to the source code.

1.2.1 **MEASUREMENT** - parse through source code and generate a variety of metrics.

1.2.2. **INVENTORY/ANALYSIS** - parse through source code to locate, identify and analyze existing system components.

1.2.3 **RE-DOCUMENTATION** - parse through source code and build system documentation.

1.3 **CONDITIONING** - automate the process of improving (changing) the code itself by passing through the source code, sometimes using a special purpose repository, to change the code. Tools of this class can be used as a preconditioning step to Repository Load/Enhancement (Tool Class 2.0)

1.3.1 **DATA RATIONALIZATION** - analyze data structures and data usage within one or more programs and support the adherence to standards and removal of redundancy (homonyms and alias)

1.3.2 **PROCESS RATIONALIZATION** - analyze process isolation, reuse, and modularization characteristics with programs and support the changes to these structural components.

1.3.3 **CONVERSION/MIGRATION - translate (change) source code for languages, databases, and teleprocessing environments.**

1.3.4 **CODE (PROCESS) RESTRUCTURING - parse source code, analyze the control flow, and correct the programs structure.**

2.0 REPOSITORY LOAD/ENHANCEMENT - parse source code for both data and processes and translate the code into the information models of a target tool repository.

2.1 DATA

2.2 PROCESS

3.0 NEW/REPLACEMENT SYSTEMS - support the development of new and replacement systems using the Information Engineering or some other life cycle methodology. This class is comprised of the sub classes 3.1 Planning, 3.2 Analysis and Design, 3.3 Construction/Generation, 3.4 GUI Builders, 3.5 Prototyping/Simulation, and 3.9 MetaCASE Tools. Tools for modeling workflows and business processes are part of Planning (Class 3.1).

4.0 **REPOSITORIES** - the tool repositories and data dictionaries which facilitate the reengineering (replacement) of existing systems or the new development of replacement systems. These are the repositories found as part of tools in Classes 1.1.2 Maintenance Environment, 2.0 Repository Load/Enhancement, and 3.0 New/Replacement Systems.

4.1 TOOL REPOSITORIES/DATA DICTIONARIES - the specific repositories found in tool classes 1.1.2, 2.0, and 3.0

4.2 REPOSITORY/DATA DICTIONARY MANAGERS - support the definition of many information models, can establish and maintain the objects comprising these models, and can manage the reuse of objects.

4.3 DATA WAREHOUSES

72

5.0 **INTEGRATED TOOLSET ENVIRONMENT** - those tools that allow for the integration of different tools into an environment, allow two or more tools to exchange data or pass control information, and those tools from a single vendor which span multiple tool classes.

5.1 **INTEGRATION FRAMEWORKS** - tools which serve as an integrated project support environment (IPSE) for integrating multiple tools into a common environment.

5.2 **INTEGRATION UTILITIES** - tools supporting the transfer of data between two or more tools or two or more tool repositories.

5.3 **RESULTING INTEGRATED TOOLS** - those tools typically from different vendors in Classes 1.0 through 4.0 and 6.0 through 8.0 which are integrated by frameworks or utilities.

5.4 **ICASE TOOLS** - a toolset typically from one vendor.

73

6.0 TESTING/VALIDATION - those tools that ensure a system operates as expected or as defined by system requirements.

- 6.1 TEST PLANNING AND MANAGEMENT
- 6.2 TEST DATA GENERATION
- 6.3 EXECUTION/TESTING
- 6.4 CAPTURE/PLAYBACK
- 6.5 COVERAGE ANALYSIS
- 6.6 VALIDATION/CORRECTION
- 6.7 CODE/DATA COMPARISON
- 6.8 GUI TESTERS

7.0 SOFTWARE/PROJECT MANAGEMENT

- 7.1 PROCESS WORKBENCHES AND WORKFLOW MANAGERS
- 7.2 PROCESS METHODOLOGIES
- 7.3 PROJECT MANAGEMENT
- 7.4 ESTIMATION/PROJECTION
- 7.5 JOB ACCOUNTING/CHARGEBACK
- 7.6 PERFORMANCE MANAGEMENT
- 7.7 PROBLEM TRACKING
- 7.8 CONFIGURATION/CHANGE/VERSION MANAGEMENT
- 7.9 DOCUMENT MANAGEMENT AND IMAGING
- 7.10 REQUIREMENTS MANAGEMENT
- 7.11 OPERATIONS MANAGEMENT
- 7.12 TRAINING
- 7.13 ACQUISITION/CONTRACT MANAGEMENT

8.0 DBMS/NETWORK/FILE MANAGEMENT

0.0 MISCELLANEOUS

76

Template Software

Enterprise Solutions with Objects

Randy Maroney, VP Bus Dev
maroney@ template.com, http://www.template.com

TEMPLATE SOFTWARE

RKM 1 8/7/95

SNAP is Targeted to Mission-Critical Applications

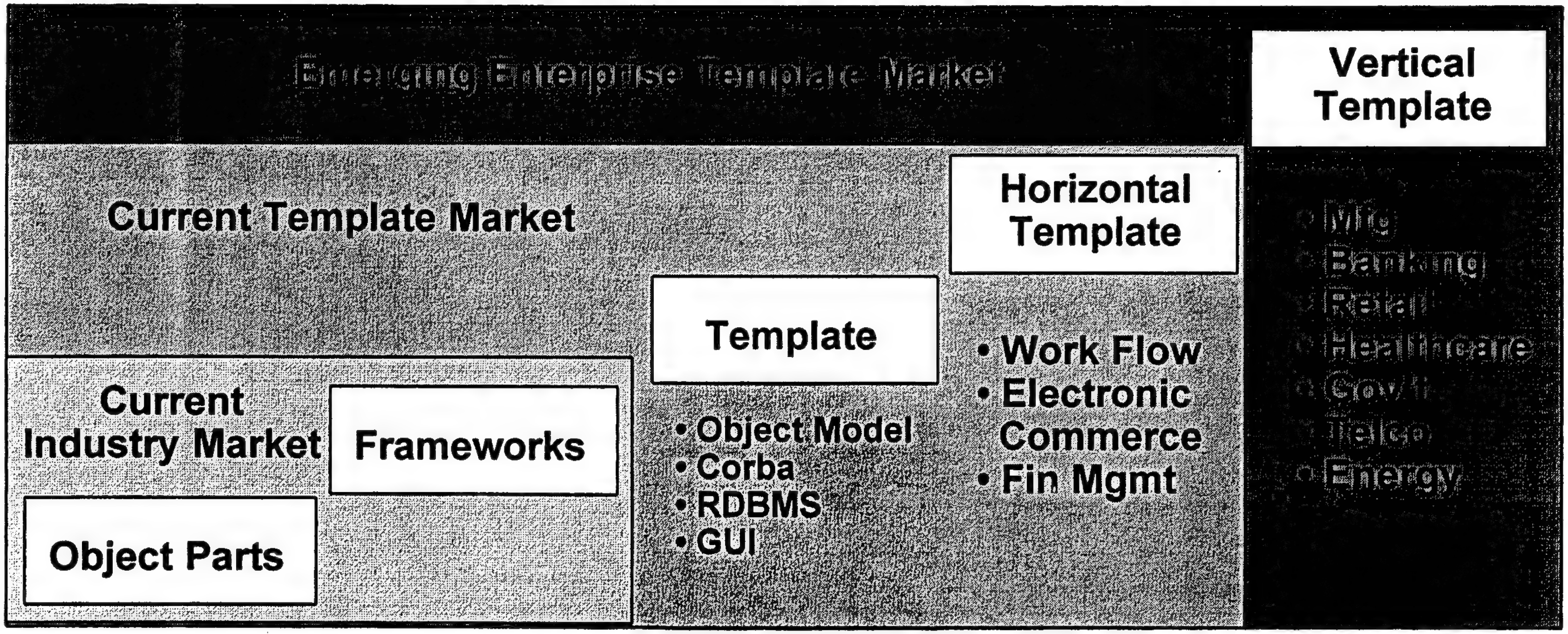
Sample of Clients	Industry	Fielded Applications
ENRON	Energy	Gas pipeline management
GTE/Spacenet	Telecommunications	TV monitor, control & broadcast scheduling
NSR/EPRI	Energy	Power grid restoration
Pemex	Petrochemical	Order fulfillment and distribution
UPS	Transportation	Aircraft maintenance logistics management
Westinghouse	Transportation	Air traffic control (flight planning)
Nafin Bank	Banking	Currency trading
EDS	Telecommunications	Credit card fraud detection
ValMex	Financial Securities	Front & back office trading

SNAP's Distributed Object Technology has been used to field 60 operational systems in over 200 companies

TEMPLATE SOFTWARE

RKM 2 8/7/95

OO Technology Evolution



- C++ Lib
- 3GL

- APIs
- Programmer

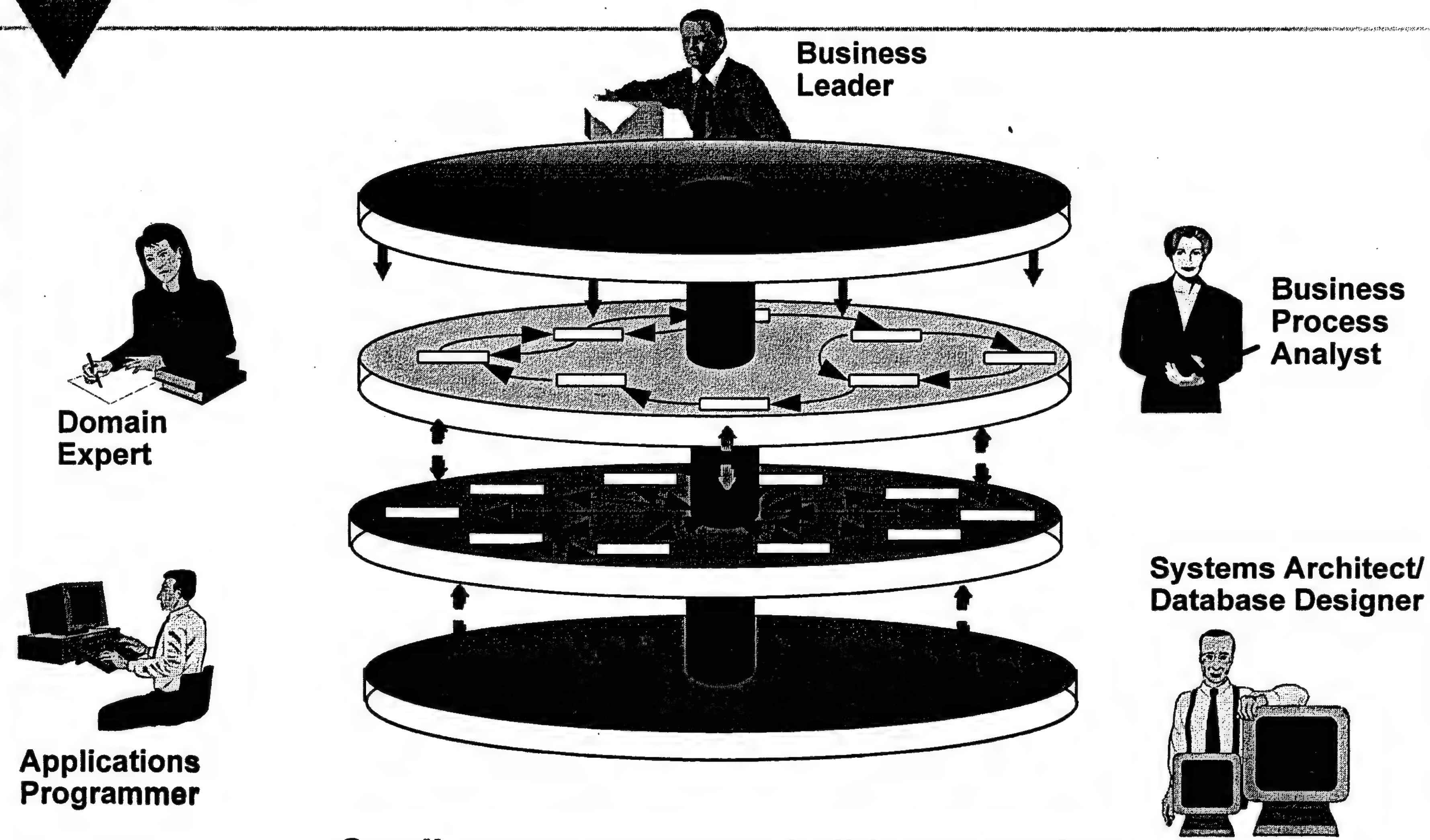
- Integrated Architecture
- Visual Application Gen
- Plug & Play Frameworks
- Cross Industry Reuse

- Industry Specific
- Enterprise FWs
- Business Models

TEMPLATE SOFTWARE

RKM 3 8/7/95

Mobilizing Skills From Across the Enterprise



Small, expert teams can build systems fast using the Template Technology Strategy

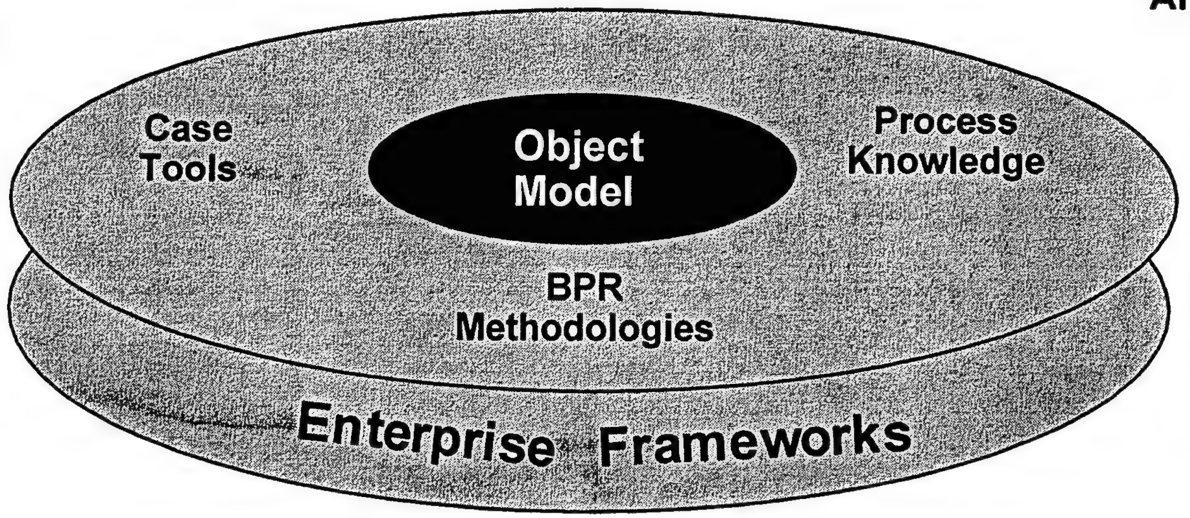
TEMPLATE SOFTWARE

RKM 4 8/7/95

Business Knowledge Layer in the Template Environment

Business
Process
Reengineering

Business
Process
Analysis



Business
Operation
Management

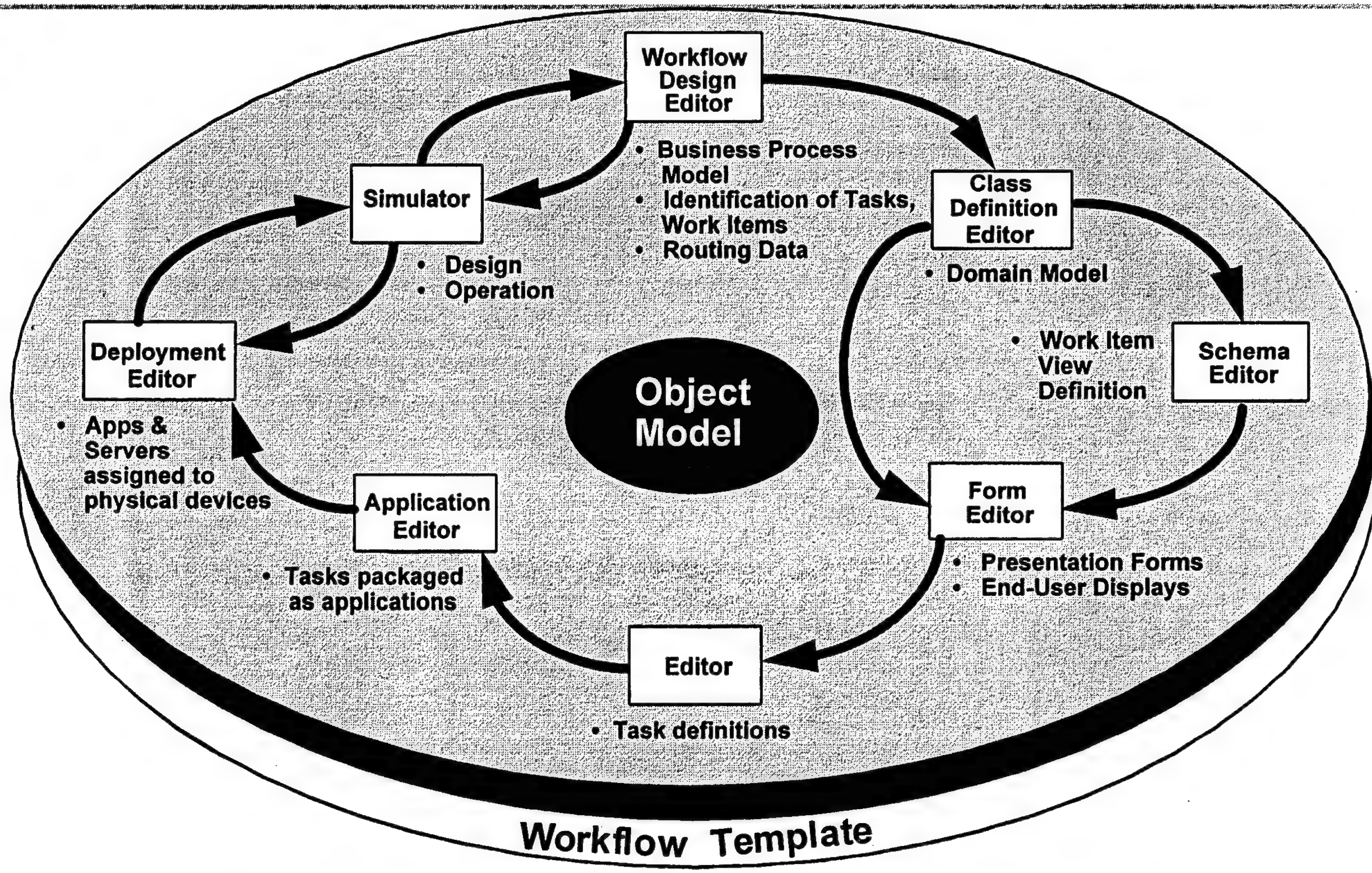
Domain
Expert

Template supports knowledge input from many sources

RKM 5 8/7/95

TEMPLATE SOFTWARE

Visual Business Programming Layer

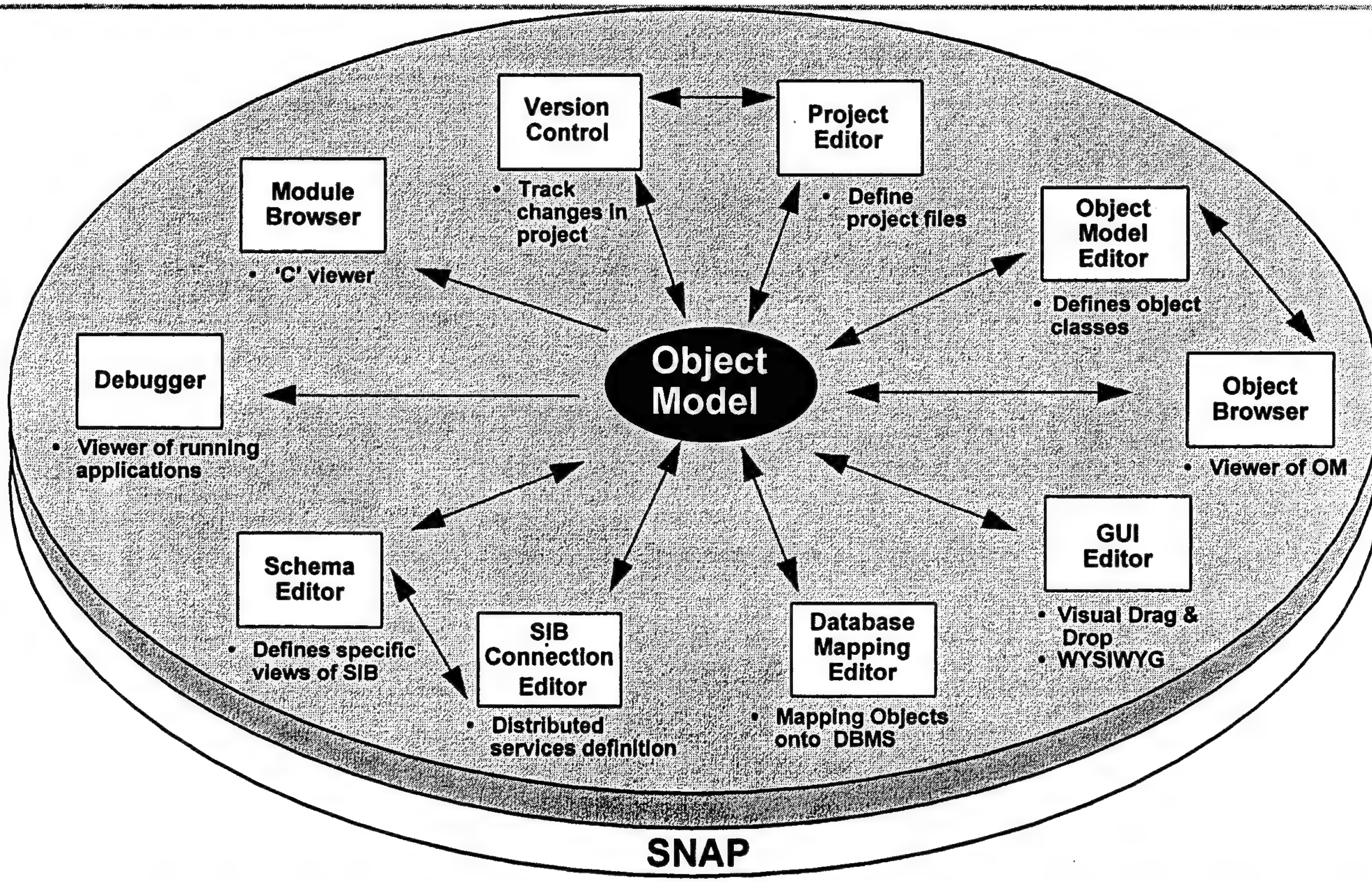


The system can change with the business needs, typically in 4-8 week iterative cycles

TEMPLATE SOFTWARE

RKM 6 8/7/96

Object Programming Layer in the Template Environment

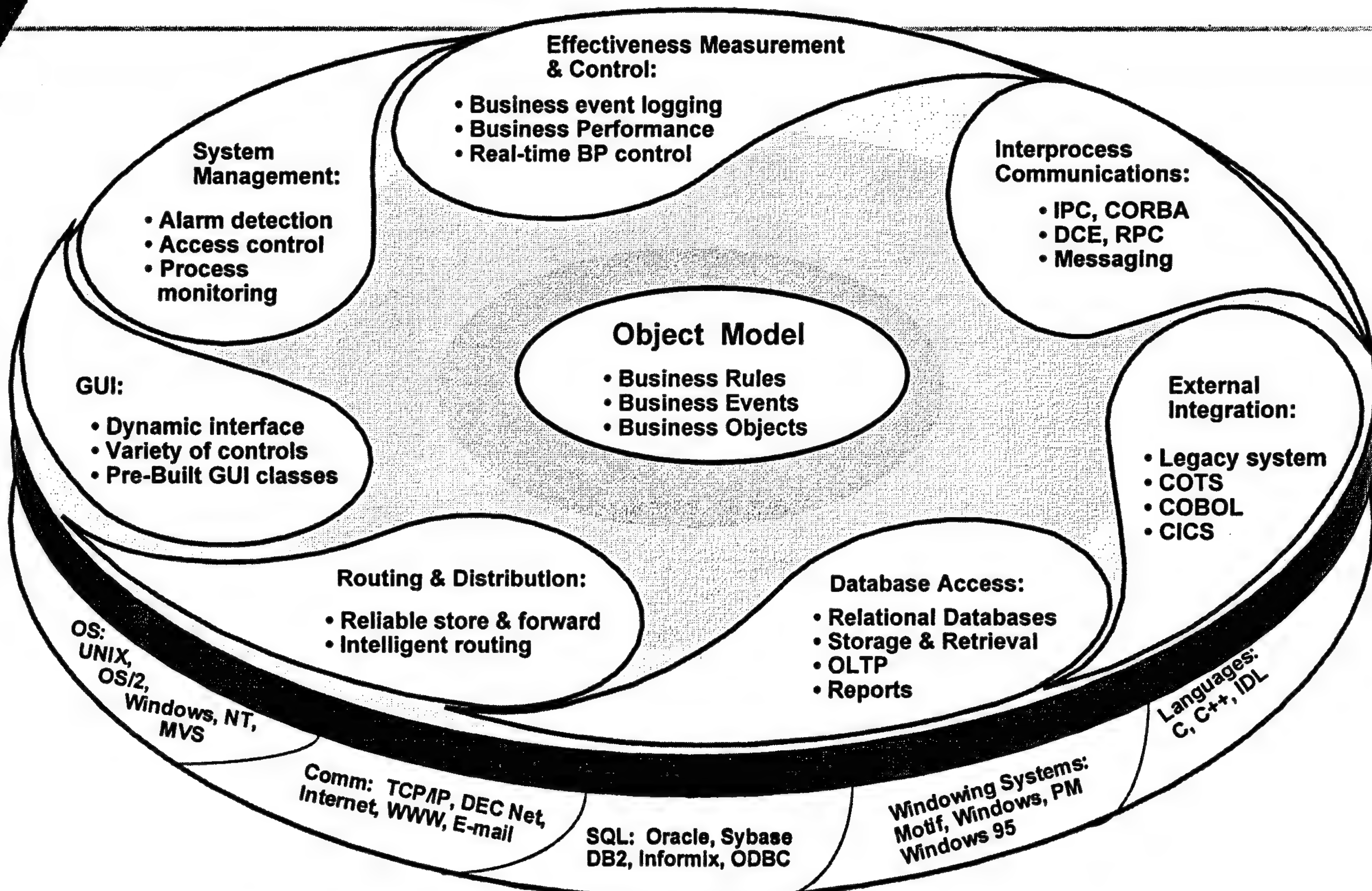


SNAP provides a full set of project development tools centered on the object model ensuring ultra-high productivity

TEMPLATE SOFTWARE

RKM 7 8/7/95

Infrastructure Layer in the Template Environment

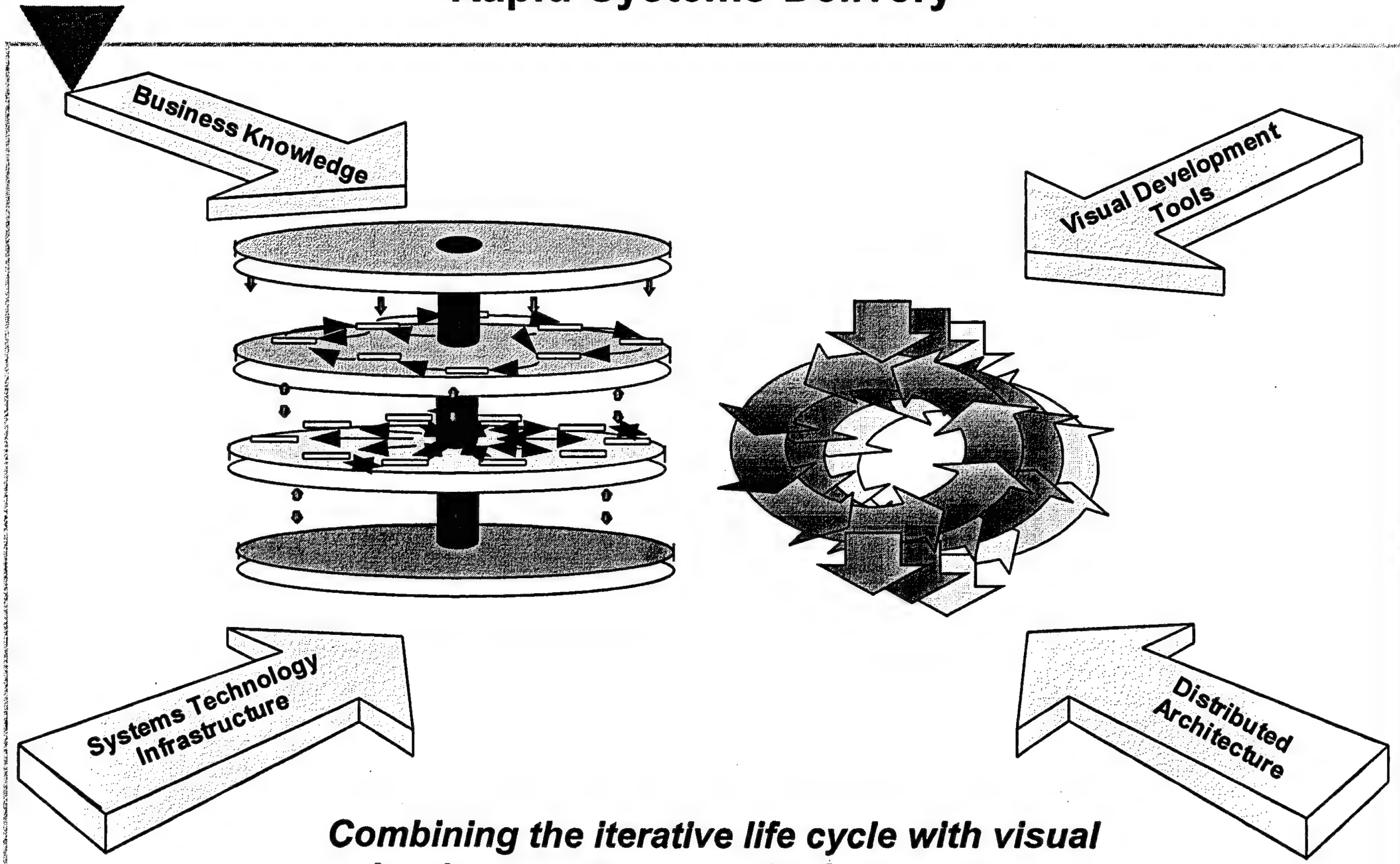


Specialized business processes and objects are portable and scaleable across the infrastructure

TEMPLATE SOFTWARE

RKM 8 8/7/95

Rapid Systems Delivery



Combining the iterative life cycle with visual development tools results faster delivery

TEMPLATE SOFTWARE

RKM 9 8/7/95

Electronic Commerce - The Second Wave

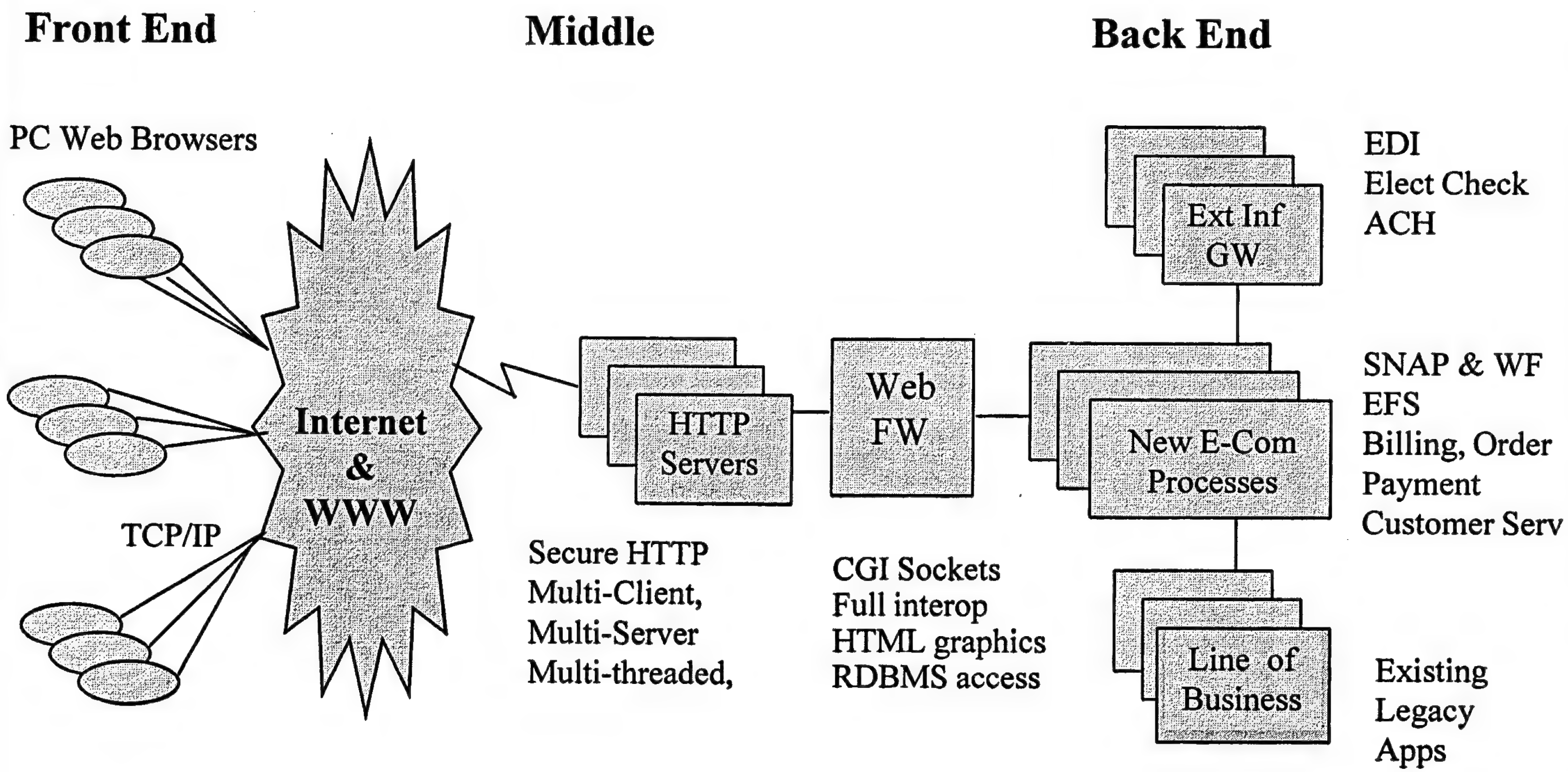
- Internet and WWW applications can be divided into three waves
  - Enterprise Communications
  - Business-to-Business Commerce
  - Consumer Commerce
- Each wave has its own set of enabling technologies and challenges
- Each wave successively builds upon and extends the baseline of the preceeding wave

Electronic Commerce begins with the business and grows

TEMPLATE SOFTWARE

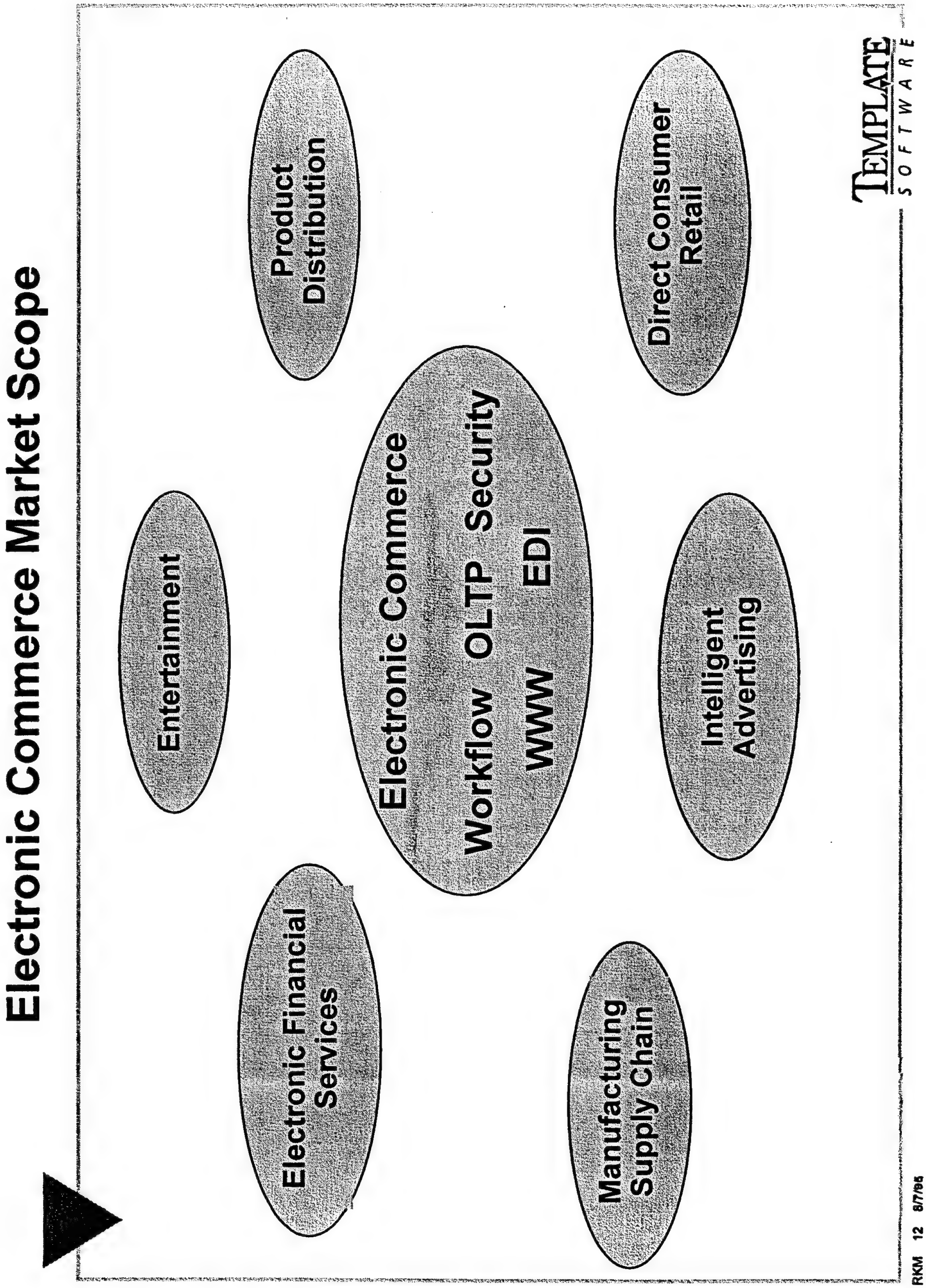
RKM 10 8/7/95

Online Internet Architecture



TEMPLATE SOFTWARE

RKM 11 8/7/96


Electronic Commerce Market Scope
Electronic Financial Services
Entertainment
Product Distribution
Electronic Commerce
Workflow OLTP Security
WWW EDI
Manufacturing Supply Chain
Intelligent Advertising
Direct Consumer Retail
TEMPLATE SOFTWARE
RKM 12 8/7/95

Using Electronic Commerce in New Markets

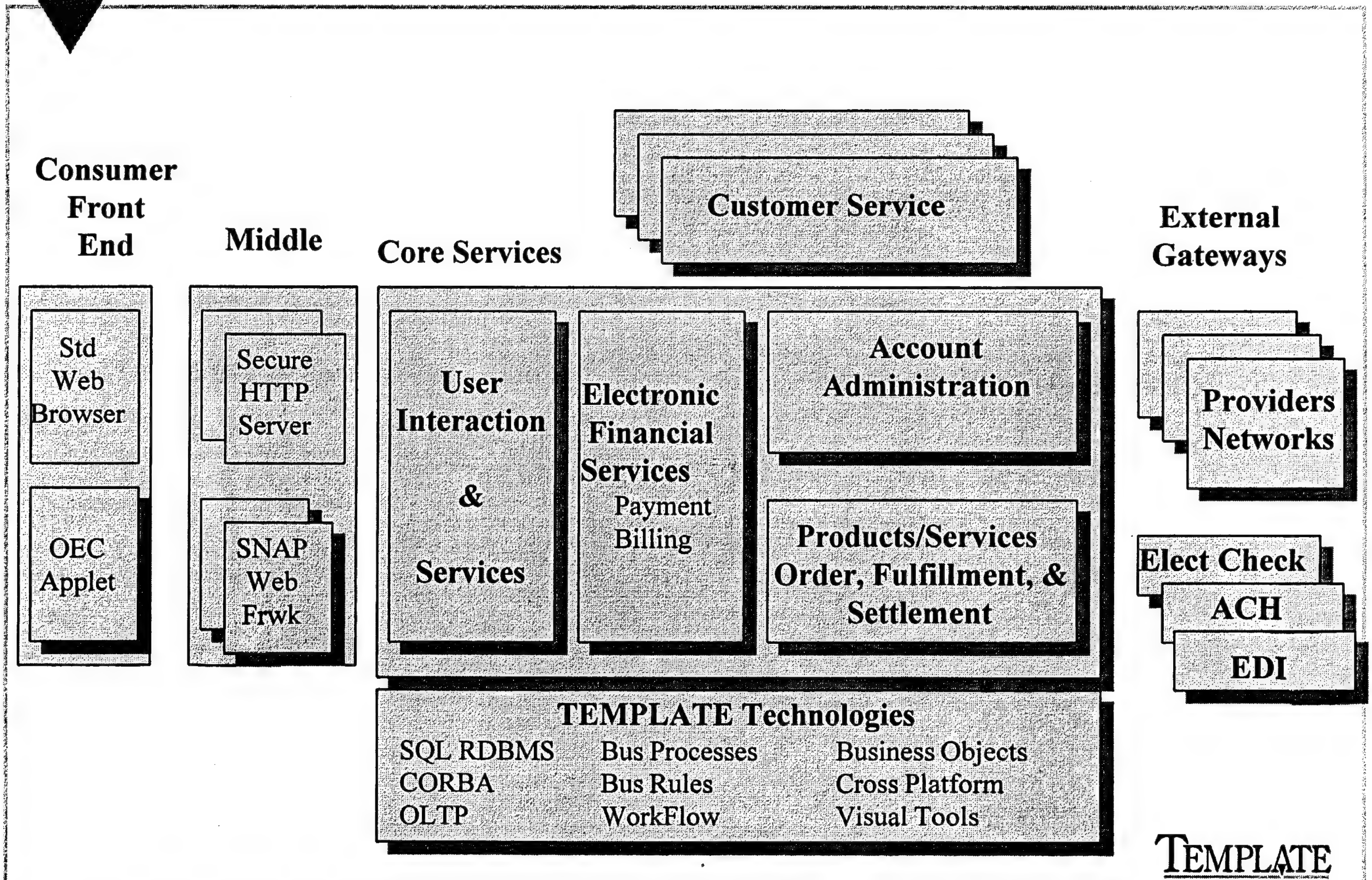
- Business to Business
- Business to Consumer
- Consumer to Business
- All business transactions
  - Orders and subscriptions
  - Billing and Payments
  - Supplier Interaction
  - Distribution Providers
  - Customer Account Services
  - Field Service
  - Electronic Advertising

- PC access via Web
- Full Security
  - Privacy
  - Access
  - Non-repudiation
- Interactive
- Integrated with Legacy Line of Business Apps
- Integrated with Electronic Financial services

RKM 13 8/7/95

TEMPLATE SOFTWARE

Electronic Commerce Application Architecture



RKM 14 8/7/95

Template '96 WWW Facilities

- Web Framework - any SNAP or WF process directly interactive with remote browser over Web
- Generation of HTML on the fly
- Web remote RDBMS query through Web FW
- Interoperable with any HTTP server or Browser
- Use industry-leading, de facto stds for security
  - RSA encryption & public/private keys
  - Finance-specific stds - ACH, Electronic Check
- Leading to "Just-In-Time" on demand applet software distribution to extend std Web browsers

TEMPLATE SOFTWARE

RKM 15 8/7/95

Summary

- Template and Customers are delivering large scale, high impact solutions
- Solutions leverage existing Template technology
- Object Technology has proven to be an effective reengineering approach
- Reengineering defines the Enterprise Business Objects
- Reengineering defines the Business Processes
- Reengineering and Template reuse architecture allows rapid, cost effective solution delivery

RKM 16 8/7/95

TEMPLATE SOFTWARE

Carnegie Mellon University
Software Engineering Institute

Towards a Framework for Program Understanding

Scott R. Tilley
stilley@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University

SPC '95

December 5, 1995

1

Carnegie Mellon University
Software Engineering Institute

Outline

Outline

1. Introduction
2. Cognitive aspects
3. Canonical components
4. Taxonomy
5. Summary

2

Motivation

- Goal: Evolutionary development
- Problem: Legacy systems
- Approach: Reengineering
  - Engineering: Constrained problem solving
  - System: Full-spectrum decision analysis
  - Software: Program understanding (PU)
  - Managerial: Project management
  - Economic: Return on investment

3

Framework

- Goal: Classify PU technology
- Developed in three steps:
  1. Investigate cognitive aspects
  2. Identify canonical activities
  3. Categorize support mechanisms
- For comparison---not evaluation

4

Carnegie Mellon University
Software Engineering Institute

Cognitive aspects

Cognitive aspects

- **Multiple problem factors**
- **Numerous cognitive models**
- **Program understanding:**
  - **Focuses on artifacts & relationships**
  - **Requires inverse domain mapping**
  - **Aided by *reverse engineering***

Carnegie Mellon University
Software Engineering Institute

Canonical components

Steps

- **Model: Construct domain-specific models of the application**
- **Extract: Gather raw data from the subject system**
- **Abstract: Create abstractions that facilitate understanding**

Carnegie Mellon University
Software Engineering Institute

...Canonical components

Artifacts

- Data: Factual information used as basis for study & reasoning
- Knowledge: The sum of what is known or derived
- Information: Selectively communicated knowledge

Carnegie Mellon University
Software Engineering Institute

...Canonical components

Activities

- Data gathering
- Knowledge organization
- Information exploration

Taxonomy

- Domain retargetability
- Scalability
- Automation level

- Pattern abstraction level
  - Program analysis
  - Plan recognition
  - Concept assignment

Carnegie Mellon University
Software Engineering Institute

...Taxonomy

- Toolset extensibility
- Cognitive support
- Application domain
- Interaction method

11

Carnegie Mellon University
Software Engineering Institute

...Taxonomy

- Standards support
- Modeling support
- Adoption cost
- Understanding-in-the-many support

12

Carnegie Mellon University
Software Engineering Institute

Summary

Summary

- PU classification framework
- Aid users in:
  - Evaluating claims
  - Assessing applicability
  - Comparing approaches
- Single perspective on reengineering

13

Carnegie Mellon University
Software Engineering Institute

...Summary

Future work

- Refine taxonomy
- Populate framework
- Perform experiments

14

This page intentionally left blank.

SOFTWARE PRODUCTIVITY CONSORTIUM

Reverse Engineering of Code into Requirements Specifications

Mark R. Blackburn

SOFTWARE PRODUCTIVITY CONSORTIUM

Outline

- Problem
- Objectives
- Benefits
- Approach
- Related research investigations
- Next steps

SOFTWARE PRODUCTIVITY CONSORTIUM

Problem Context

- Testing to support reengineering can account for 50-75% of the cost [Sneed95]
- Common reengineering approach is to:
  - Reengineer legacy into an equivalent system
  - Use legacy system as an oracle for testing the new system
  - Evolve newly reengineered system
- Difficult to develop test sets to ensure that the desired functionality of the legacy exists in new system
- Need basis for developing tests
  - Requirements provide basis, but difficult to extract from legacy

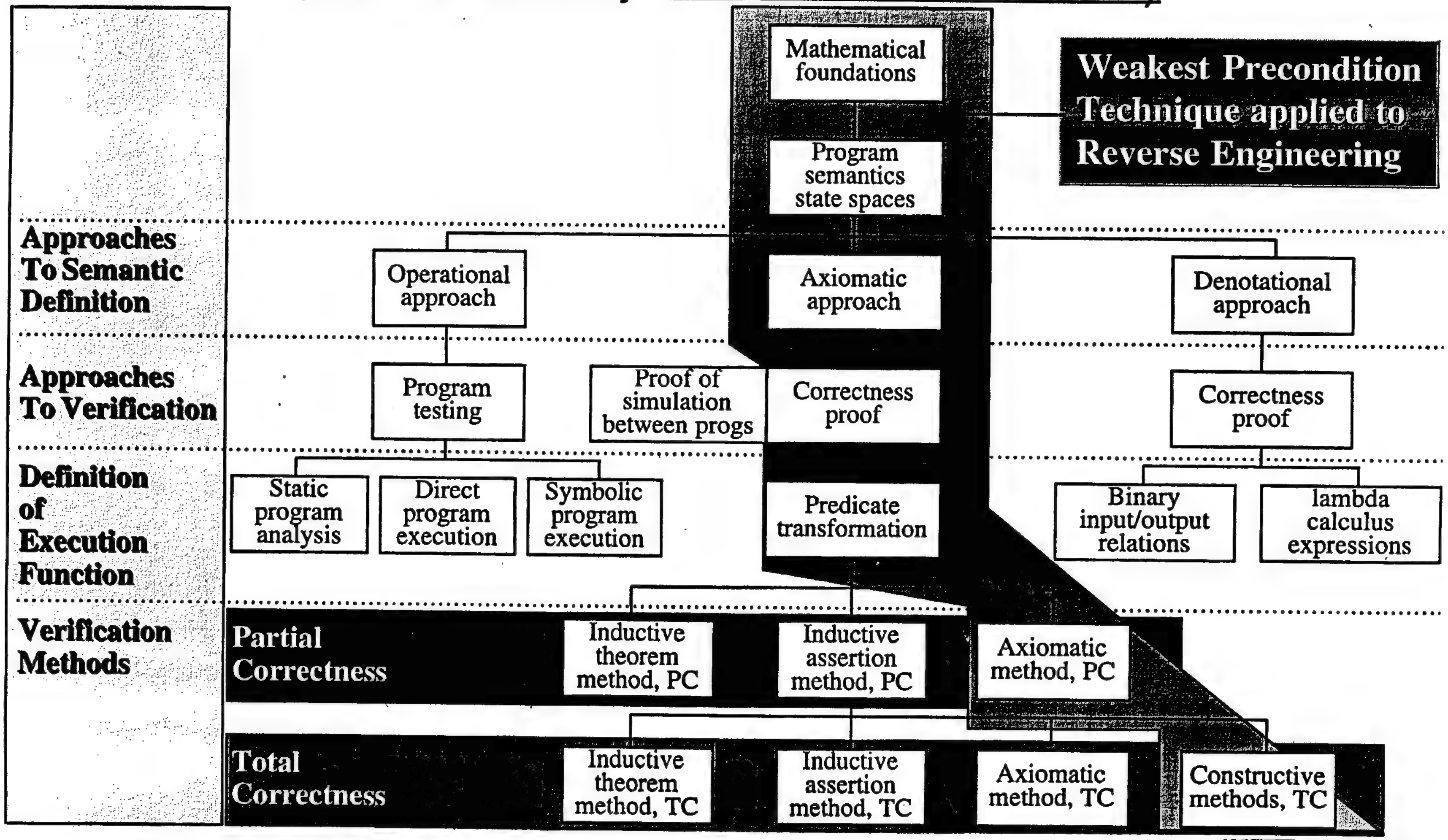
Objectives

- Reverse engineer requirement specifications from code
- Derive a formal requirement specification based on strongest precondition model
  - Extend weakest precondition technique [Dij76] described by Pizzarello and Hart [Piz95, Har95]
- Develop heuristics models to support representation of domain concepts and transformation rules for mapping code to requirements

SOFTWARE PRODUCTIVITY CONSORTIUM

Context: Formal Perspective

(from Taxonomy of Verification Methods)



Berg et. al.1982

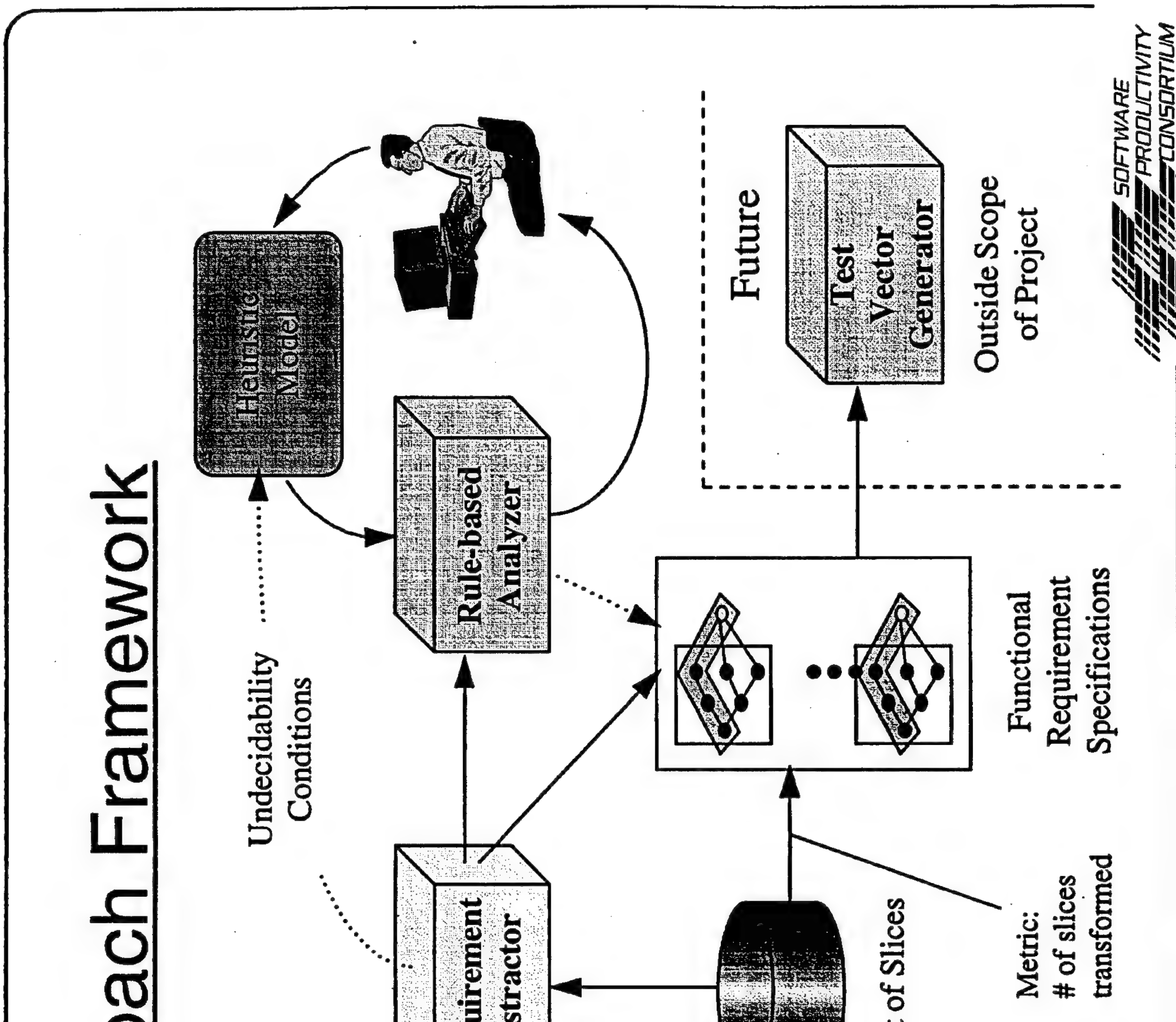
Benefits

- Requirements could be used to automatically generate test cases to assess the reengineered system
- Requirements would support the transformation, and evolution of the reengineered system
- Concept can be applied to other problems
  - Development of high assurance software [NISTa]

SOFTWARE PRODUCTIVITY CONSORTIUM

Related Research Investigations

- Most reverse engineering approaches use analysis techniques derived from **operational approach**
- Reverse transformational approach based on **denotational approach** to characterization of program [War93]
- Constructive method associated with **axiomatic approach**
  - UNITY at UT Austin
  - Peritus Software using UNITY
- Reverse engineering using heuristic models based on domain analysis
  - Representation is key
  - Need inferencing capabilities integrated with model representation
  - Need pattern matching to search for abstract constructs


Approach Framework
Requirement Specification Model
Slice criterion
Slicer
Code File
Code File
Set of Slices
Requirement Abstractor
Undecidability Conditions
Heuristic Model
Rule-based Analyzer
Functional Requirement Specifications
Metric: # of slices transformed
Future
Test Vector Generator
Outside Scope of Project
SOFTWARE PRODUCTIVITY CONSORTIUM
Copyright © 1995, Software Productivity Consortium, Inc. All rights reserved.

Next Steps

- Use SNAP for rule-based analyzer
  - Allows domain concepts to be represented in an object model
- Provides two unique attributes for modeling heuristics
  - Supports inference rules as part of object model
  - Supports pattern/language analysis and representation
- Inference rules are used in the transformation process
- Patterns/language supports recognition of language relationships that can be abstracted together or away

SOFTWARE PRODUCTIVITY CONSORTIUM

This page intentionally left blank.

Peritus™
Software Services, Inc

PROGRAM MAINTENANCE TECHNIQUES EXPERIENCE WITH LOGICAL CODE ANALYSIS IN SOFTWARE MAINTENANCE, REUSE AND RE-ENGINEERING

John Hart
Peritus Software Services, Inc.
304 Concord Rd
Billerica, MA 01821
508-670-2500
jmhart@world.std.com

daj:mh:Rev A:1 ... experts in software maintenance

Peritus™
Software Services, Inc

Three Principal Maintenance Activities

- Corrective Maintenance
  - Fixing defects in existing software
- Adaptive Maintenance
  - Changing specifications, reuse, enhancements
- Perfective Maintenance
  - Performance improvements
  - More efficient memory and file space usage
  - Improving documentation
  - Simplifying code for maintainability and reuse
- Summary: Maintenance is a challenging and costly part of the software life-cycle

daj:mh:Rev A:2 ... experts in software maintenance

Page 1

Peritus™
Software Services, Inc

What is Logical Code Analysis?

- Determination of postconditions and weakest preconditions to determine code properties
  - Requires simple predicate calculus
- Emphasis is on logical properties of the code
  - Not its operational behavior
- Sequential code only (for now)
- Basic theoretical technique
  - Dijkstra's weakest precondition (*wp*)
    - Other work by Gries, Cohen, ...
    - Newer work on parallel programs
      - Chandy & Misra, ..

daj:mh:Rev A:5 ... experts in software maintenance

Peritus™
Software Services, Inc.

Experience with Logical Analysis

- Our success is due to:
  - Analyze code rather than derive (synthesize) code
  - Apply analysis to code slices that affect a limited number of variables
  - Isolate small code segments likely to be the root cause of a defect or limitation
  - Annotate code as we analyze it, thus capturing knowledge
  - Analyze conditional statements: convoluted logic is the root cause of many problems
  - Our goals are modest
    - Solve some problems, increase productivity, ...

daj:mh:Rev A:6 ... experts in software maintenance

Page 3

Peritus™
Software Services, Inc

First Simple Example

- Defect report
  - "An unexpected `err_typeK` event occurred"
    - Close to a half million lines
    - A simple search using UNIX text tools showed one place where the event variable was assigned to `err_typeK`

```
/*  event is initialized to 0 */
if (read_sensor(A1) ||
   !read_sensor(A2) ||
    read_sensor(B1) ||
   !read_sensor(B2) )
       if (read_sensor(B1) )
               event = err_typeK;
```

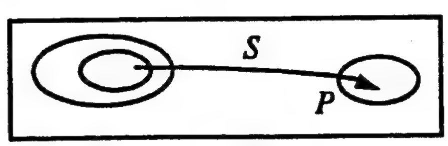
dia:jmh:Rev A:9 ... experts in software maintenance

Peritus™
Software Services, Inc

Logical Analysis Example

- Compute the "state" that results in this value
- Compute weakest precondition (*wp*)
  - Using standard techniques
  - Weakest precondition is a logical predicate function
    - The first argument is a program
    - The second argument is the postcondition
  - *wp* gives the minimal precondition to yield the postcondition
    - *wp* is the necessary initialization to get the result
  - Write

$$wp.S.P = Q$$



dia:jmh:Rev A:10 ... experts in software maintenance

Peritus
Software Services, Inc

Some Observations

- Warning - The developer originally included these tests for a reason
- Operational analysis (using dumps, debuggers, test data, and so on) might not detect the defect
- Risky assumption: `read_sensor ()` may have internal state, or it could change between the two reads
- Defect resolution required both analysis and product knowledge
- We say that this defect showed a case of *redundant testing*. *Dead code* is similar.

dajmh:Rev A:13 ... experts in software maintenance

Peritus
Software Services, Inc

Conditional Code: An Example

- Complex conditional code from a TCP/IP implementation
- Complexity is due to the complexity of the code
- Defect reports:
  - "Losing packets when the system is heavily loaded"
  - Difficult to reproduce
  - Passes the test cases (100% path coverage)
- A frequently called function is suspicious
  - Processes incoming packets
  - Short but poorly documented

dajmh:Rev A:14 ... experts in software maintenance

Peritus
Software Services, Inc

First Step to Correct the Code

- a, b, and c define a valid "window" of packet sequence numbers
  - b must be between a and and c, with a less than c
  - a and c cannot be "too far apart"
  - Introduce a parameter w defining the "window size"
- The first `if` statement is suspicious
  - Involves bit operations and comparisons
  - Bit expression comparison to 0 is TRUE exactly when a and c have the same sign
  - Rewrite first `if` statement as:

```
if ((a < 0 && c < 0)
    || (a >= 0 && c >= 0)
    || (a <= 0 && c >= 0))
```

dia:jmh:Rev A:17 ... experts in software maintenance

Peritus
Software Services, Inc

Compute *WP*

- `valid_window` returns either 0 or 1
- Result depends only on values of a, b, and c

```
wp.(r = valid_window (a, b, c)).(r == 0 || r == 1) = TRUE
```

evaluates to:

```
  wp.(r = valid_window (a, b, c)).(r == 1)
=       ((a < 0 || c >= 0) && a <= b && b <= c)
   ||   (!(a < 0 || c >= 0) && (b >= a || b <= c))
```

dia:jmh:Rev A:18 ... experts in software maintenance

Peritus™
Software Services, Inc

Observations

- valid_window accepts values of [a, b, c] such as [0, 32768, 65536]
  - Test data did not cover this case
- Create a parameter "W" to represent "window size"
  - A small power of 2, i.e., 32 or 64
  - Corresponds to the size of an array holding data packets
- Summary
  - Process was not operational
  - Some intuition and product knowledge, but logic was used most to create an exact specification - almost identical to the code
  - New knowledge can be added as annotation
  - Code is simpler

da:jmh:Rev A21 ... experts in software maintenance

Peritus™
Software Services, Inc

The Problem with `if-then-else`

- Loop did not terminate when it should

  while (!done) { Loop Body }

Loop body is:

```
if (Mode == 0) ok = TRUE else ok = FALSE;
if (!ok) {
        if (Mode == Lo) {
                if (ES < QS)  ok = TRUE;  }
        if (Mode == Hi {
                if (ES > QS)   ok = TRUE; }
}
done = FALSE;
```

(continued)

da:jmh:Rev A22 ... experts in software maintenance

Peritus Software Services, Inc.

Analyzing the `if-then-else` (2)

{ Use the identities: (p || !p) and q == (p || q) }

```
=  ok
   || ((Mode == Lo) && (ES < QS))
   || ((Mode == Hi) && (ES > QS))
```

Combining these results:

```
wp.(Lines 1-20).ok
 =     (Mode == 0)
  || ((Mode == Lo) && (ES < QS))
  || ((Mode == Hi) && (ES > QS))

/*  1-7:New Code) */
ok = (Mode == 0)
    || ((Mode == Lo) && (ES < QS))
    || ((Mode == Hi) && (ES > QS));
```

cia:jmh:Rev A25 ... experts in software maintenance

Peritus Software Services, Inc.

Analyzing the `if-then-else` (3)

```
wp.(Lines 8-16).done
    =  ok &&
           ((eMode == Lo) && (ES < MS)
      ||    (eMode == Hi) && (ES > MS))

  /* 8-16:New Code */
   done = ok &&
           ((eMode == Lo) && (ES < MS)
       ||   (eMode == Hi) && (ES > MS));
```

- Latent defect
  - » done cannot be set when `ES` is equal to either `MS` or `CS`.
    - ◆ This could require a modification changed to `<=` and `>=`
- `if-then-else` statements can be dangerous
  - » Especially when compounded

cia:jmh:Rev A26 ... experts in software maintenance

Peritus Software Services, Inc

Strategies (3)

- Strategy 3. Show that a Program is Correct
- Compute the weakest precondition to get
  - `wp.S.P == Q`
  - If Q is identically TRUE, the program is correct
  - If Q is not identically TRUE, then the program will fail with any test data that makes Q FALSE.
- State variables must be initialized so as to make Q be TRUE

da:jmh:Rev A29 ... experts in software maintenance

Peritus Software Services, Inc

Strategies (4)

- Strategy 4. Show Two Programs are Equivalent
  - We may rewrite a sequence of code to improve it in some way, even though we do not want to change its behavior
  - Common during code reengineering
    - Simpler, faster, or more maintainable
  - `wp.S.Q = wp.S'.Q`
    - for all predicates, Q

da:jmh:Rev A30 ... experts in software maintenance

Page 15

Peritus
Software Services, Inc

```
1.       if CI < 1              /*  FR != NY */
2.          if OEC == FR
3-4.           if NEC == FR    { /* Do Nothing */ }
5-6.        else               { CI = 1; MSG = 1; }
7.       else
8.          if OEC == NY
9-10.          if NEC == NY    { /* Do Nothing */ }
11-12.         else            { CI = 1; MSG = 1; }
13.         else         /* if OEC == NY */
14.            if NEC == FR)
15-16.            if (OEC == FR) { /* Do Nothing */ }
17-18.            else         { C = 1; MSG = 1; }
19.            else
20.               if (NEC == NY)
21-22.               if (OEC == NY) {/* Do Nothing */}
23-24.               else     { CI = 1; MSG = 1; }
```


Peritus
Software Services, Inc

`if-then-else` Statements & Reuse (2)

- Using Logical Analysis, we get the equivalent:

```
if (CI < 1 && OEC != NEC &&
     (    OEC == FR || OEC == NY
       || NEC == FR || NEC == NY   ))
                { CI = 1; MSG = 1; }
```

- In words, CI and MSG are set to 1 exactly when:
  - `CI` is less than **1**,
  - `OEC` and `NEC` are different, and
  - at least one of `OEC` and `NEC` is `FR` or `NY`

Peritus
Software Services, Inc

Loops

- Must determine a loop invariant
  - An invariant predicate at both beginning and end of the loop body. Invariant may be paramaterized by a loop index
  - General scheme

```
/*  General form of a loop                    */
/*  B is a logical "guard"                    */
/*  S is the loop body                        */
/*  I is an invariant of the loop             */
/** Initialize so that I is TRUE              */
while (B) {              /**    I && B        */
   S                     /**    I             */
}                        /**    I && !B       */
```

dia:jmh:Rev A:37 ... experts in software maintenance

Peritus
Software Services, Inc

Loops (2)

- Correctness requires

```
I && B ==> wp.S.I
```

- Must be identically `TRUE` if the program is to be correct
- If it is not identically `TRUE`, state values that make it `FALSE` will help to determine test points that will cause defects

dia:jmh:Rev A:38 ... experts in software maintenance

Peritus
Software Services, Inc

Detecting Loop Defects (2)

- Computing the postcondition now shows that `i == N`

```
I && !B
 =        0 <= k < j ==> A[k] < A[j]
 &&  0 <= j < k < N ==> A[j] >= A[k] && 0 <= j < i = N
 =   The program specification
```

dajmh:Rev A:41 ... experts in software maintenance

Peritus
Software Services, Inc

Detecting Loop Defects (3)

- Test 2: Is the invariant initialized? The answer is no
  - `j` and `i` are the same.
  - Initializing `i` to `1` quickly fixes this problem
- Test 3: Does the invariant implication hold?
  - Compute and simplify the implication. Is it identically TRUE?
  - Let S denote the loop body. Evaluate:

    `I && B ==> wp.S.I` This should be identically TRUE
  - In this case, it is not. `FALSE` when `a[i] == a[i-1] && j == i`
  - Giving the last bug (the comparison) and correct program

dajmh:Rev A:42 ... experts in software maintenance

Peritus
Software Services, Inc.

Example

```
1.1   i = 0;
1.2   while (i < MTRT/HZ {
1.3       /* Every time the loop body executes, either
1.4          1) There is a delay of time HZ, or
1.5          2) We exit the loop                          */
2.0       if (LP(dev)) {
3.1          if (!LPW(dev))
3.2                break;
4.0       } else
5.0       delay (HZ);
6.1       i = i + 1;
6.2   }
7.1       /* i >= MTRT/HZ or we exited loop at line 3.2 */
7.2   if (i >= MTRT/HZ) error_msg (252);
```

da:jmh:Rev A:45 ... experts in software maintenance

Peritus
Software Services, Inc.

Example

- Does that help?

It might! Part of the problem is that the "break" at line 3.2 violates structured code principles. We can fix that as follows:

da:jmh:Rev A:46 ... experts in software maintenance

Peritus™
Software Services, Inc

Example

```
1.1   i = 0;
1.2   exit = FALSE;
1.3   while (i < MTRT/HZ && !exit) {
1.4       /* Every time the loop body executes, either
1.5          1) There is a delay of time HZ, or
1.6          2) We exit the loop                          */
2.1       if (LP(dev) && !LPW(dev)) {exit = TRUE}
3.1       if (LP(dev) && LPW(dev))  { }
4.1       if (!LP(dev))             {delay HZ}
6.1       i = i + 1;
6.2   }
7.1       /* i >= MTRT/HZ || exit                         */
7.2   if (!exit) error_msg(252);
```

cla:jmh:Rev A:49 ... experts in software maintenance

Peritus™
Software Services, Inc.

Example

- Now it is clear that our assumptions (as expressed in the comment) about the loop body are not valid. The fix is easy.
- NOTE: This defect occurred in some real OS code. Using operational techniques, the defect was unresolved for a long time.

cla:jmh:Rev A:50 ... experts in software maintenance

Peritus™
Software Services, Inc

Future Work

- Continuous improvement of our training materials and methodologies
- Development of logical analysis tools to be part of software toolkits
- Extending the application of our techniques to concurrent programs

dajmh:Rev A53

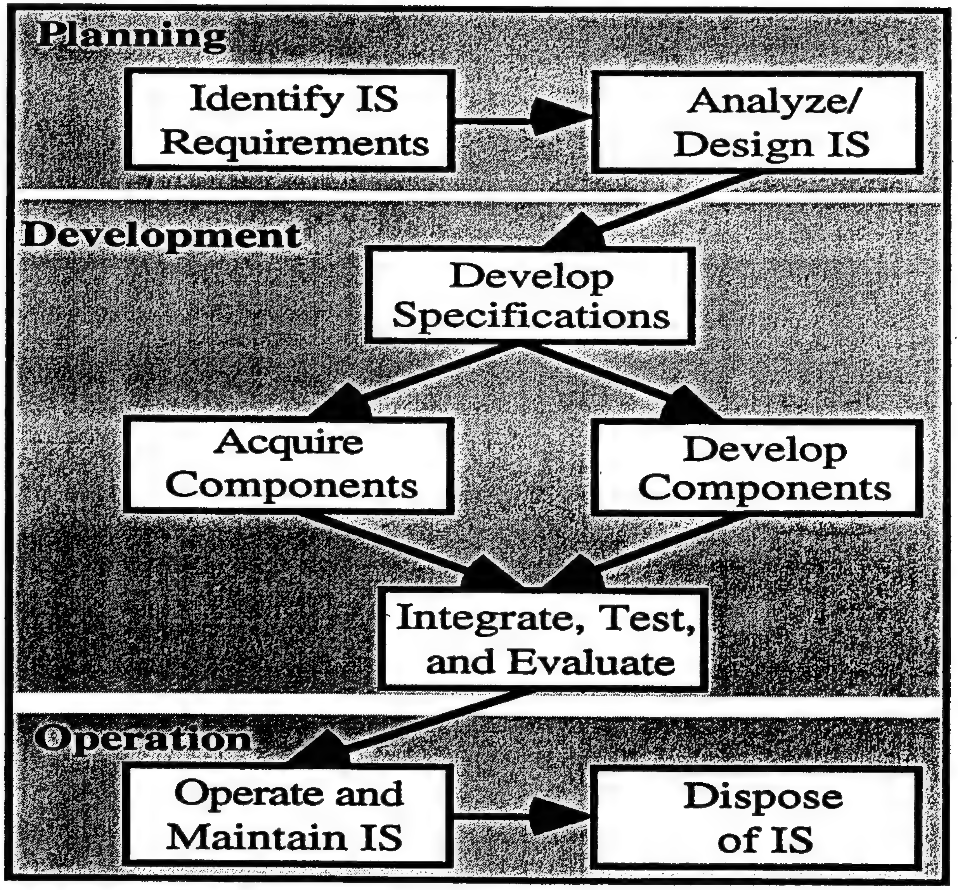
... experts in software maintenance

From Business Reengineering to Information Systems

Clem McGowan

5 December 1995

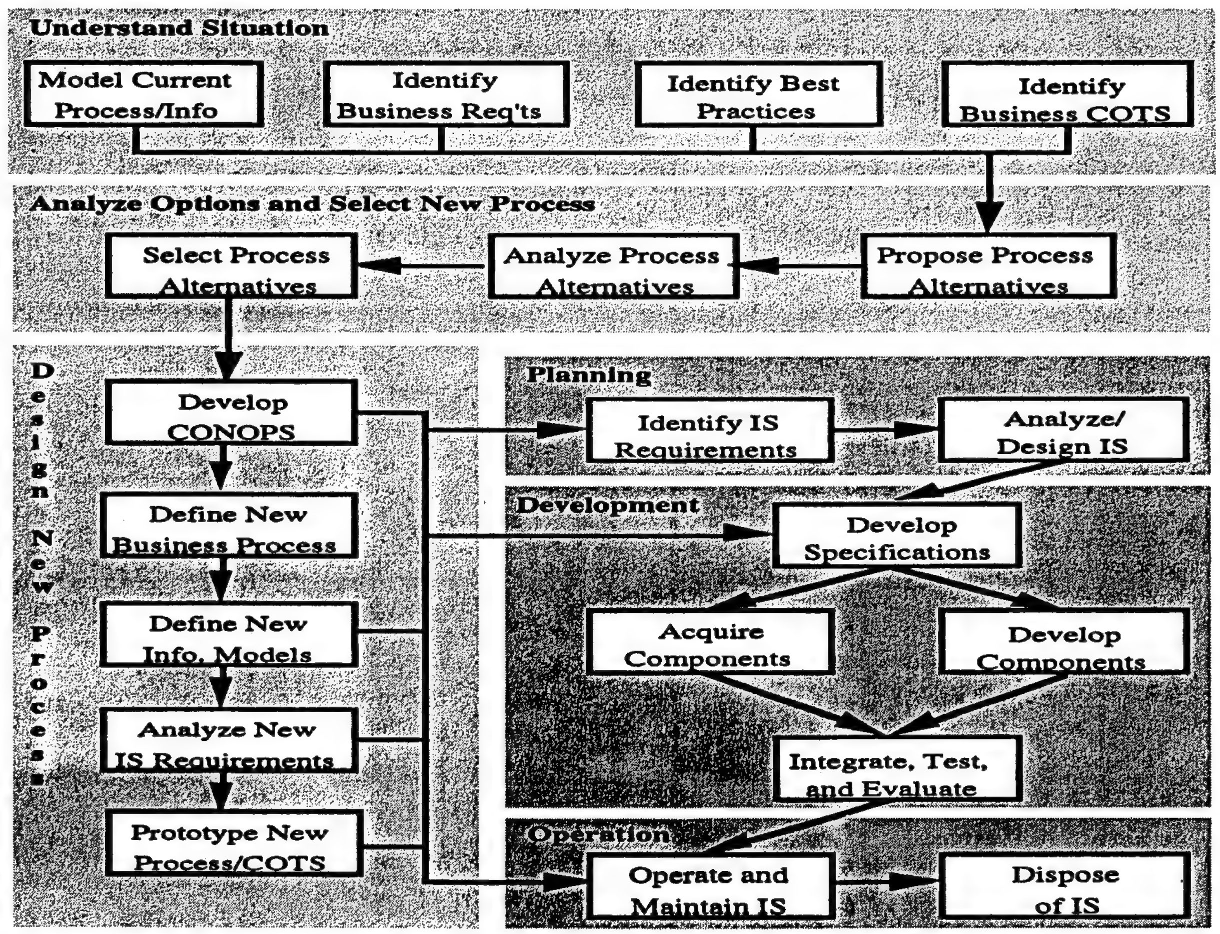
Standard Information System Engineering (ISE) Process



IS = Information Systems

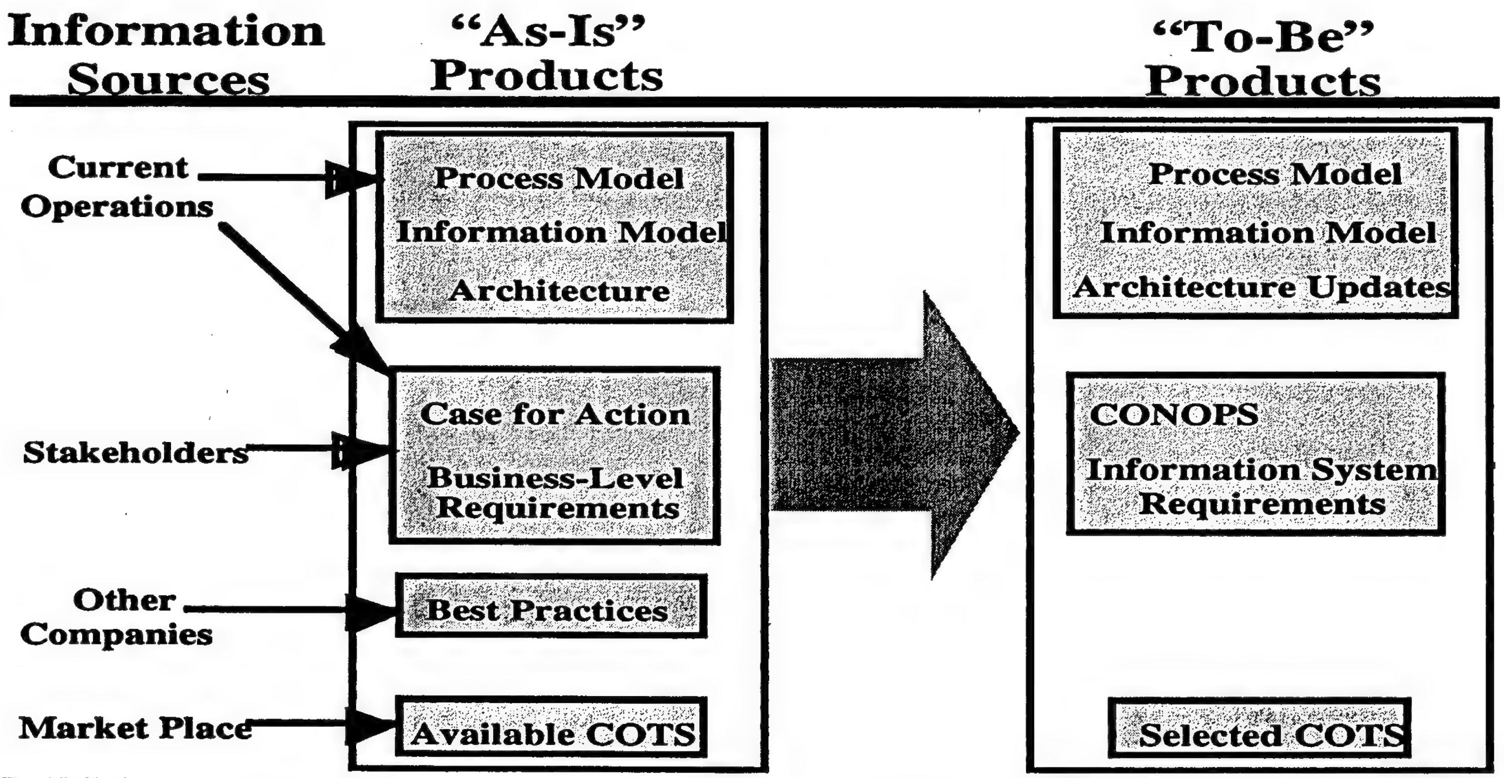
MITRE

Mapping BPR Results to ISE Process



MITRE

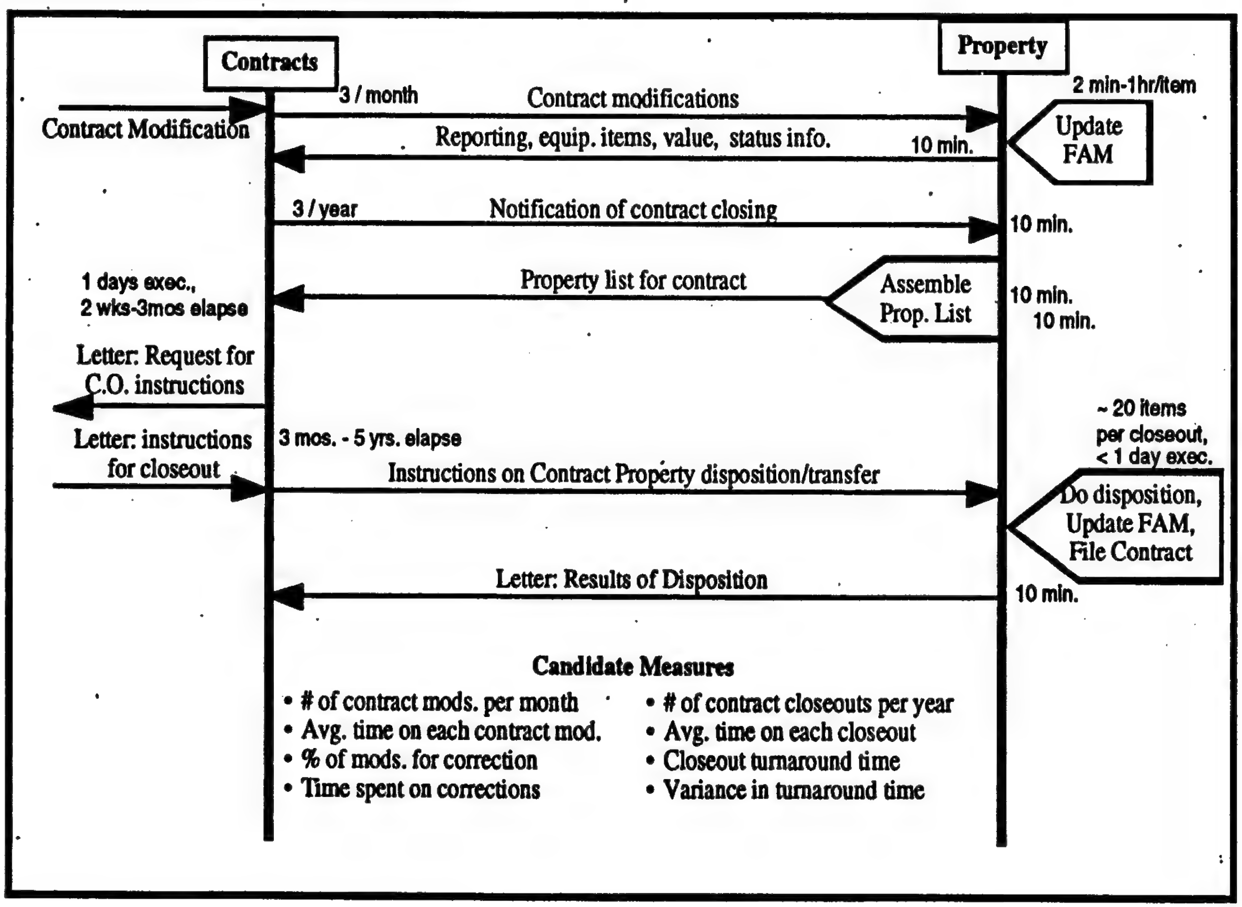
BPR Products



<u>Important Supporting Information</u>: Process Alternatives, Customer Satisfaction Data, Measures of Merit, Estimates of Improved Performance and Impact on Stakeholders, Prototype Results, Issues (Problems, Deficiencies, Opportunities)

MITRE

Interaction Diagram



MITRE

A BPR "Case for Action"

1. CASE FOR ACTION

1.1 Mission Statement

<Describes current vision and direction of the organization.>

1.2 Goals and Success Measures for Operations

<Describes how success can be measured in terms of the goals of the organization.>

1.3 Assessment of Current Operations
- **1.3.1 Meeting Mission Goals**
- **1.3.2 Efficiency in Use of Resources**
- **1.3.3 Strengths and Weaknesses**

1.4 Assessment of Current Information and Communications Systems
- **1.4.1 Aligned with Mission**
- **1.4.2 Strengths and Weaknesses**

1.5 Assessment of Future Trends and Their Implications
- **1.5.1 In Information Systems**
- **1.5.2 In Communications**
- **1.5.3 In Other Technologies**

1.6 Assessment Of High-Level Benchmarking Results
- **1.6.1 Applicable Best Practices of Other Organizations**

1.7 Gap Between Current Situation and Future Goals

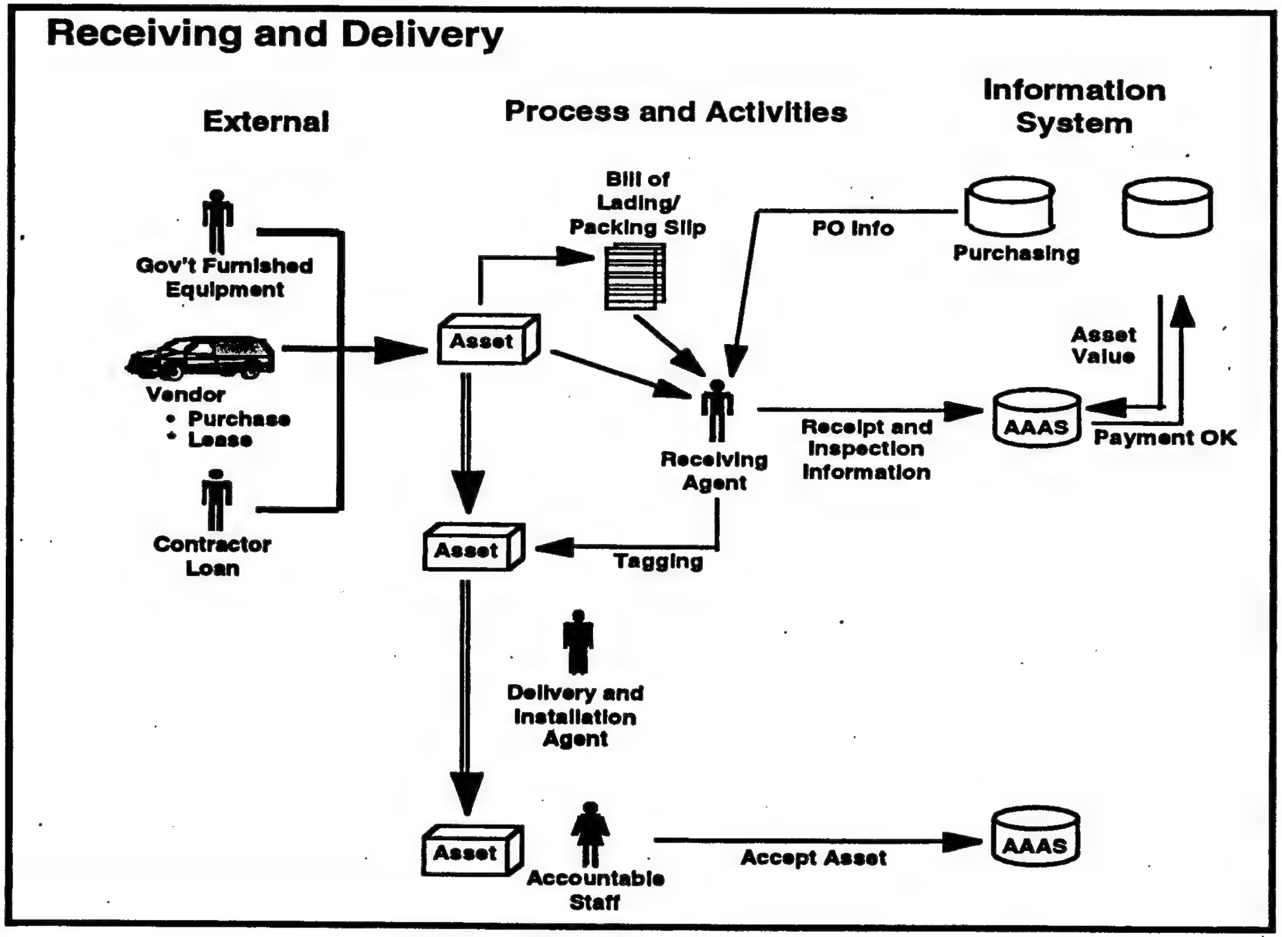
1.8 Major Challenges and Risks to Achieve Goals for Operations

1.9 Recommended Overall Approach to Achieve the Goals for Operations

1.10 Plan to Develop a Detailed Road Map
- **1.10.1 Activities and Products**
- **1.10.2 Roles and Responsibilities**
- **1.10.3 Schedule**

MITRE

Concept of Operations (high level example)



MITRE

Southwest Airlines
(manage and utilize assets)

- Only airline consistently profitable over the decade

category	Southwest	Industry Ave.
Flight/plane	11/day	7/day
pass./employee	2,400	900
employee/plane	80	125
Operating profits	17%	3%

- Avoid hub-and-spoke; avoids “peaks and valleys”
  - —volume pairs: Dallas-Houston, SF-LA, Chicago-St. Louis
  - —use cheaper, less congested, second-tier airports
  - —20 minutes turn around (95%) versus > 45 min.
  - —only 60% thru travel agents; no reserve of seating
- Only one type of plane (Boeing 737s)
  - —reduced maintenance and training costs

MITRE

10

Measures/Success Indicators to Inspire Creativity in Reengineering

New Process Goals	Some Indicators of Success (Measures of Merit)	Candidate Process and Policy changes to improve performance with respect to these indicators
Completeness of Accountable Asset Coverage	• % correct in regular, internal audits • % correct and time to complete formal audits • confidence of sponsors and of management in asset system	— conduct rolling internal audits to assess — make individuals accountable for assets and responsible for maintaining asset status in AAAS — regular discussions with sponsors and management (reporting on status of asset management action list)

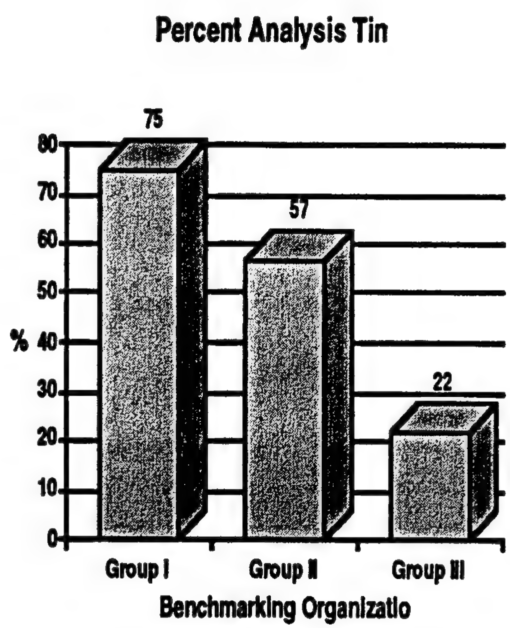
MITRE

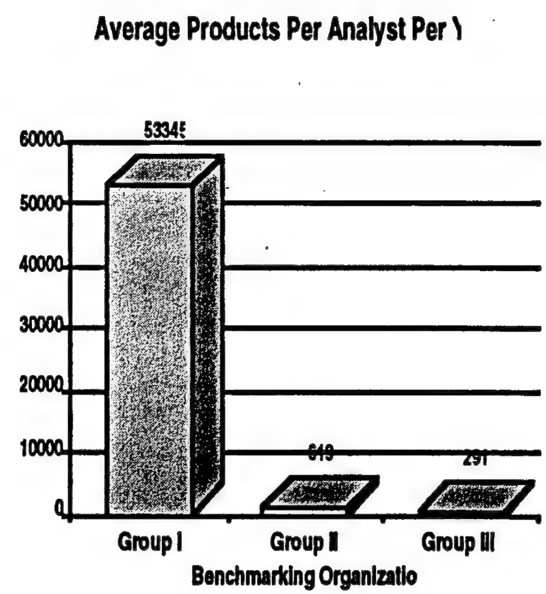
Some Candidate Measures

Type of Measurement	Examples of this type
Input	Total orders, workload, materials and their costs, investments , and the cost of money
Process	Four distinct processing categories: direct operations, management activity, quality assurance, and transportation/ communication. For each process activity in each category measure the fixed and variable costs and time delays.
Output	Throughput (i.e., amounts of products produced, services rendered and sales made), Number of successful transactions, total operational expenses, inventory (including work-in-process) and product quality (e.g., error density)
Outcome	Cycle time (from order to delivery), Customer Satisfaction, Market Share, and Profits, Resource Utilization
Derived	Efficiency, Effectiveness, Productivity, Unit Costs, Flexibility in offerings, Levels of Service, Cost of Quality, and Return on Management (Tooth-to-Tail Ratio)

MITRE

Benchmarking (at high process level) Suggests New Process Possibilities





MITRE

13

Classes of Requirements — Format/Tools

Class of Requirements	Format/Tools
Business Process Work Flow High Level Requirements, Needs, Problems Policies and Procedures Staff Training	Narrative (CONOPS) with Schematic Diagrams Text statements IDEF Process & Information Flows Data Model
Information and Data	Information and Data Models (MITRE Enterprise Data Model using Oracle CASE)
Functional: Detailed/Derived Requirements and Business Rules Functional Descriptions: Functions, Transactions, Scenarios	Text statements Automated Tools to manage and analyze requirements (e.g. RTM, DOORS, IEF, ADW) Program Design Language (PDL) or Data Flow Diagrams
Non-functional: - Performance - Capacity - Reliability - Accuracy - Security	Text statements Automated Tools to manage and analyze requirements (e.g. RTM, DOORS, IEF, ADW)

MITRE

14

Determine How Changes Will "disturb" Different Classes of Stakeholders

User Class	Change to System	Differences	Possible Resistance
Employee	•Assigned individual responsibility for assets. •Changes in asset status entered into System by responsible employee.	•Responsible employee accepts and relinquishes responsibility of assets in a controlled manner.	•Employees will feel additional burden of asset responsibility. They may require more security control on their assigned assets (see security). •Employees will need to be trained on new asset movement, transfer, and disposal procedures (training material may be on-line). ...

MITRE

Activity Based Costing (ABC) and Functional Economic Analysis

- Activity model of the process
- Associate fixed and variable costs with the activities
- Analyze workload impact on resources and measures
- Compare against an ABC model for the alternative process
- Leads to a better and different assignment of costs to product and services (than overhead distribution)
- e.g., MITRE Purchasing with the following measures
    - PR processing delay versus time
    - Staffing versus time
    - Audit backlog versus time
    - Cash flow versus time

MITRE

Project Results (w.r.t. Measurable Goals) for Different Scenarios

System Lifetime Years	Annual PR $ Growth	Discount Rate 0%			Discount Rate 7%		
		Annual Growth In Number Of PRs					
		-5%	5%	15%	-5%	5%	15%
5	0%	15.70	16.59	17.83	12.53	13.24	14.21
	5%	17.95	18.84	20.08	14.31	15.02	15.99
10	0%	35.16	38.74	45.86	24.09	26.37	30.71
	5%	44.91	48.49	55.61	30.23	32.50	36.85

MITRE

Managing a BPR-to-IS Requirements Project

BPR Activity / WBS Task	Wor k Units
A0 – Reengineer Business	1000
A1 – Manage Process Reengineering Effort	150
A11 – Produce and Approve BPR Project Plan	20
A12 – Build and Direct BPR Team	30
A121 – Establish BPR Team Requirements	8
A122 – Identify BPR Expertise	3
A123 – Identify Business Domain Expertise	3
A124 – Identify Information System Expertise	3
A125 – Identify Candidate Project Leaders	3
A126 – Acquire Team Member Resources	10
A13 – Get Buy-in from Process Stakeholders	40
A131 – Determine Key Business Process to Reengineer	5

MITRE

More Project Activities [≈ 20% resources to understand current process]

A2 - Understand Current Business Process Situation	200
A21 - Interview Stakeholders	50
A211 - Schedule Interviews with Key Process Stakeholders	5
A212 - Assemble Material for Interviews	5
A213 - Conduct Interviews with Stakeholders	25
A214 - Organize Notes from Interviews	15
A22 - Capture Business Level Requirements	40
A221 - Analyze Existing Requirements	15
A222 - Review Stakeholder Interview Notes	5
A223 - Identify New Requirements	10
A224 - Record Business Requirements	10
A23 - Identify "Best Practices"	30

MITRE

Different Expertise Needed in Different Stages of a Project

Expertise/Knowledge	Activities	Description
Domain Organization	A13, A15, A21, A41	Knowledge of organizational structure, key decision-makers, process stakeholders (and interfaces between them), and organizational policies and procedures.
Business Process	A25, A32, A33, A41, A42	Can describe and evaluate performance of key business processes of the organization. Identifies resources, customer interfaces, responsibilities, and constraints.
Best Practice	A23, A32	Knowledge of current best practices in the application domain. Contacts in competing or similar businesses.
Business Information	A25, A42	Understanding of information produced by the business process and information needed to run the business.
Business COTS	A24, A31, A32, A45	Knowledge of commercial-off-the-shelf software applications used in the business domain.
Business Operations	A25, A41	Experience in business areas where process is applied as well as interactions with related business areas.

MITRE

4. RESULTS

At the end of the workshop, each participant was asked to list the topics that interested her or him. The topics were collected and presented to the participants. Each participant was then asked to vote on the three topics that most interested her or him. The following table lists the topics and the number of votes assigned to each topic.

Topic	Number of Votes	Topic	Number of Votes
Potholes and Pitfalls of Reengineering	3	Code Translation	3
Tools Experience	9	User Interface Reengineering	6
BPR and Software Engineering Relationship	6	Metrics	3
Systems Reengineering	8	"Wrapped" Legacy System Reengineering	0
Transition Planning	11	Product Lines	4
COTS	9	How Much	2
Planning	13	Methods—OO	3
Methods—CS	3	Reengineering Cost/Benefit Analysis	14

Furthermore, the following World Wide Web sites were identified as useful sources of reengineering information:

- www.afmc.wpafb.af.mil
- www.reengineer.org/forum
- www.sei.cmu.edu/~reengineering
- www.softwre.org

This page intentionally left blank.

APPENDIX A. WORKSHOP ATTENDEES

The following is an alphabetical list of all attendees of the Second Software Productivity Consortium Reengineering Workshop.

Name	Organization	Address
Aiken, Peter	Virginia Commonwealth University	(804) 828-0174 paiken@caball.vcu.edu
Bohan, Jennifer	Vitro Corporation	(703) 418-8275 bohanj@vitro.com
Chikofsky, Elliot	DMR Group, Inc.	(617) 272-0049 e.chikofsky@computer.org
Clark, John	Comptek Federal Systems, Inc. Va. Beach Engineering Services	(804) 463-8500 x316 clark@comptek.com
Davis, Ted	Software Productivity Consortium Reuse & Reengineering Project	(703) 742-7335 davis@software.org
Evers, Ed	CACI CACI Advanced Technology Center	(703) 841-7838 eevers@hq.caci.com
Facemire, Jeff	Software Productivity Consortium Reuse & Reengineering Project	(703) 742-7189 facemire@software.org
Fee, Sandra J.	Vitro Corporation Software Center of Excellence	(301) 231-1403 fee@vitro.com
Feerrar, Caterine M.	Vitro Corporation AUA-11	(301) 738-5314 feerrar@vitro.com
Gerritsen, Douglas	U. S. Army Armaments R&D Center	(201) 724-3587 dgerrit@pica.army.mil
Gracza, John W.	CASE Associates, Inc.	(703) 978-7120 cai@teleport.com
Graves, Robert	Vitro Corporation Advanced Software Technology	(301) 231-3126 gravesr@vitro.com

Name	Organization	Address
Greene, Robert	Lockheed Martin Corporation EPI Center	(609) 338-3465 greene@esc.camden.mmc.com
Hart, Johnson M.	Peritus Software Services	(508) 670-2500 x223 jmhart@world.std.com
Johnson, Robert E.	Single Agency Manager (SAM)	(703) 697-5397 robert.johnson@comm.hq.af.mil
Kromholz, Alfred	Software Productivity Consortium Reuse & Reengineering Project	(703) 742-7274 kromholz@software.org
Linger, Rick	Software Engineering Institute	(301) 926-4858 rlinger@sei.cmu.edu
Martin, Pauline F.	Vitro Corporation SP	(301) 231-3129 martinpf@vitro.com
McCreary, Julia	Internal Revenue Service Data Administration & Design Planning	(703) 235-2755 julia.mccreary@ccmial.irs.gov
McGowan, Clem	The MITRE Corporation	(703) 883-7099 mcgowan@mitre.org
Mutafelija, Boris	Northrop Grumman Data Systems & Services Division	(703) 713-4174 borism@gateway.grumman.com
O'Grady, Jim	GDE Systems Advanced Engineering Techology	(619) 592-5079 ogrady@gdesystems.com
Phisterer, Cathy	CACI IT Company - Federal	(703) 277-6768 cphisterere@std.caci.com
Rose, Anne	University of Maryland Human-Computer Interaction Lab	(301) 405-2757 rose@cs.umd.edu
Sharon, David	CASE Associates, Inc.	(503) 656-0986 cai@teleport.com
Sisson, Philip	Lockheed Martin	(703) 264-6433 sisson.phil@ist.vf.mmc.com
Sutherland, David	Lockheed Martin Corporation Information Systems Company	(407) 826-7956
Tilley, Scott R.	Software Engineering Institute Carnegie Mellon University	(712) 268-7157 stilley@sei.cmu.edu
Ulery, Bradford T.	The MITRE Corporation	(703) 883-3313

Name	Organization	Address
	Software Engineering Center	bulery@mitre.org
West, Stacy	Vitro Corporation SP	(301) 231-2543 westsl@vitro.com
Wetzel, Paul	Vitro Corporation Advanced Software Technology	(301) 231-3095 wetzel@vitro.com
White, Karen	TASC	(617) 942-2000 x2654 krwhite@tasc.com
Wilson, Mark	NSWC White Oak Naval Surface Warfare Center	(301) 394-5099 mlwilso@relay.nswc.navy.mil

This page intentionally left blank.

APPENDIX B. WORKSHOP AGENDA

The following is the final agenda of the Second Software Productivity Consortium Reengineering Workshop.

MONDAY, DECEMBER 4, 1995

8:15 - 8:35	Welcome and Introduction	J. Facemire, the Software Productivity Consortium
8:35 - 9:15	The IS Reengineering Spectrum: Terms, Approaches, Methods, Tools	E. Chikofsky, DMR Group
9:15 - 9:25	Software Productivity Consortium	A. Pyster, the Software Productivity Consortium
9:25 - 10:10	SPC's Product-Line Approach	J. Facemire, the Software Productivity Consortium
10:10 - 10:25	Break	
10:25 - 11:55	IS Reengineering Experience Reports	E. Chikofsky, DMR Group (moderator)
	- Internal Revenue Service	
	- Integrating Domain Engr. and Reengr.	J. McCreary, IRS
	- Others	N. Prywes, CCCC/U Pa
	- Discussion	
11:55 - 12:25	Reengineering of User Interfaces	J. Facemire, the Software Productivity Consortium (moderator)
	- UI Reengineering at Maryland	A. Rose, Univ. of Maryland
12:25 - 1:25	Lunch	
1:25 - 2:40	Data Reengineering	E. Chikofsky, DMR Group (moderator)
	- Focus: Data Reengineering	P. Aiken, VA Commw U
	- Discussion	
2:40 - 2:55	Break	
2:55 - 3:55	Reengineering Economics	J. Facemire, the Software Productivity Consortium (moderator)
	- Software Reengr. Assessment Handbook	J. Clark, Comptek
	- Discussion	
3:55 - 4:25	Tools for Reengineering	E. Chikofsky, DMR Group (moderator)
	- Classifying Tools for Reengineering State of the Industry	D. Sharon, CASE Assoc.
4:25 - 5:00	What We Heard: Summary Discussion of the Day (all attendees)	J. Facemire, the Software Productivity Consortium E. Chikofsky, DMR Group

TUESDAY, DECEMBER 5, 1995

8:15 - 9:15	Object Technology in Reengineering	M. Blackburn, the Software Productivity Consortium (moderator)
	- Enterprise Solutions with Objects (demo)	R. Maroney, Template SW
	- Discussion	
9:15 - 10:30	Reverse Engineering	E. Chikofsky, DMR Group (moderator)
	- Framework for Progr. Understanding	S. Tilley, SEI
	- Rev Engr. Code into Requirements (include Logical Code Analysis)	M. Blackburn, the Software Productivity Consortium
	- Discussion	J. Hart, Peritus
10:30 - 10:45	Break	
10:45 - 12:00	Reengineering Information Systems	A. Kromholz, the Software Productivity Consortium (moderator)
	- Recap: SPC's Product Line Approach	J. Facemire, the Software Productivity Consortium
12:00 - 1:45	Reengineering Information Systems	A. Kromholz, the Software Productivity Consortium (moderator)
	- Bridging Gap Betw BPR and SW Syst.	C. McGowan, MITRE
	- Discussion	
1:45 - 2:45	What We Heard and What We Need Summary Disc. and Prioritization	J. Facemire, the Software Productivity Consortium E. Chikofsky, DMR Group
2:45 - 3:00	Closing Remarks/Adjourn	J. Facemire, the Software Productivity Consortium